RIGA TECHNICAL UNIVERSITY
FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
INSTITUTE OF APPLIED COMPUTER SYSTEMS

Practice #3
“Database Management Systems”

Advanced SQL Constructs

Student Name	Student Card No.
Emir Oguz	230ADB011

Checked: Andrejs Gaidukovs

2022 / 2023 Study Year

Content

1 Task

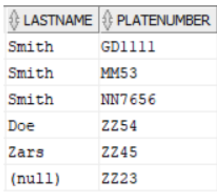
1. Fill table with students participating in Practice on the 1st page.
2. Practice 3 is based on database created in Practice 2 database.
    a. If you sucessfully implemented this database in Practice 2, just continue with queries.
    b. If not script file for database generation *2.CarRentalScriptPractice2.sql* is attached in Ortus.
3. In query 10 addition to database should be made. Script is attached in this document Query 10 task and also in ortus with filename (both are the same): *3.CarRentalScriptPractice3addingHierarcyToCustomers.sql*. Script creates unary relation in table Customers.
4. Write queries according to given task, copy text of your query and take screenshot of result (like shown in examples below).
5. Upload (one per team) in Ortus:
    a. report file in MS Word format
    b. script file in .sql format.

Example, how to fill queries part:

1.1 Query 1

Task: Extract people with cars.

```
Select p1.lastname, c1.platenumber
From Cars C1, People P1
Where p1.pid(+)=c1.pid;
```

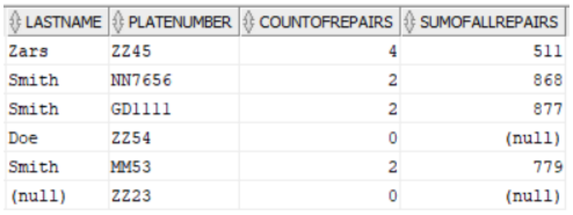


LASTNAME	PLATENUMBER
Smith	GD1111
Smith	MM53
Smith	NN7656
Doe	ZZ54
Zars	ZZ45
(null)	ZZ23

1.2 Query 2

Task: Extract cars with owners, count of repair and sum of all repairs for each car.

```
select p1.lastname, c1.platenumber, count(r1.repid) AS CountOFRepairs,
sum(r1.repsum) as SumOfAllRepairs
from people p1, cars c1, repairs r1
where (p1.pid(+)=c1.pid) and (r1.fk_carid(+)= c1.carid)
Group by c1.platenumber, p1.lastname;
```



LASTNAME	PLATENUMBER	COUNTOFREPAIRS	SUMOFALLREPAIRS
Zars	ZZ45	4	511
Smith	NN7656	2	868
Smith	GD1111	2	877
Doe	ZZ54	0	(null)
Smith	MM53	2	779
(null)	ZZ23	0	(null)

2 Database Description

2.1 Conceptual Model

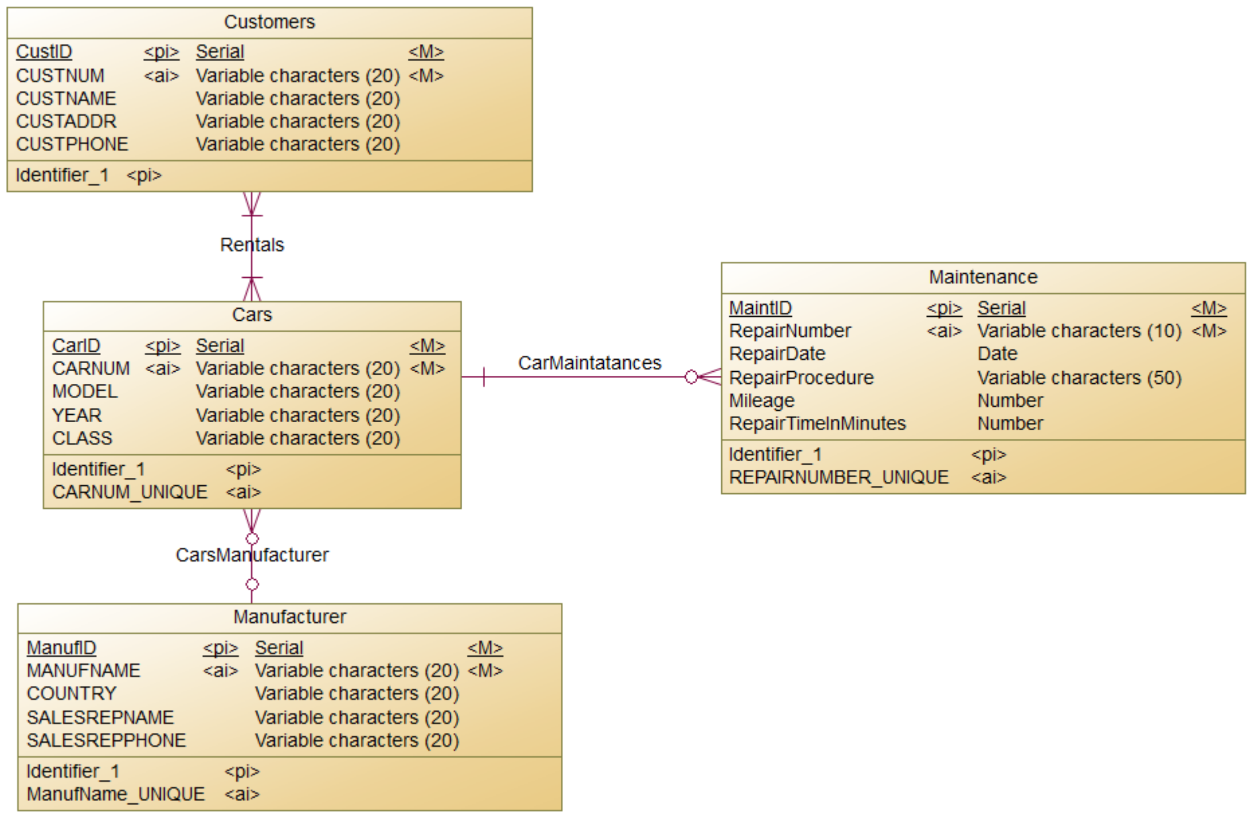

Customers
CustID <pi> Serial <M>
CUSTNUM <ai> Variable characters (20) <M>
CUSTNAME Variable characters (20)
CUSTADDR Variable characters (20)
CUSTPHONE Variable characters (20)
Identifier_1 <pi>
Rentals
Cars
CarID <pi> Serial <M>
CARNUM <ai> Variable characters (20) <M>
MODEL Variable characters (20)
YEAR Variable characters (20)
CLASS Variable characters (20)
Identifier_1 <pi>
CARNUM_UNIQUE <ai>
CarMaintatances
Maintenance
MaintID <pi> Serial <M>
RepairNumber <ai> Variable characters (10) <M>
RepairDate Date
RepairProcedure Variable characters (50)
Mileage Number
RepairTimeInMinutes Number
Identifier_1 <pi>
REPAIRNUMBER_UNIQUE <ai>
CarsManufacturer
Manufacturer
ManufID <pi> Serial <M>
MANUFNAME <ai> Variable characters (20) <M>
COUNTRY Variable characters (20)
SALESREPNAME Variable characters (20)
SALESREPPHONE Variable characters (20)
Identifier_1 <pi>
ManufName_UNIQUE <ai>

Figure 1. Conceptual Model

2.2 Physical Model

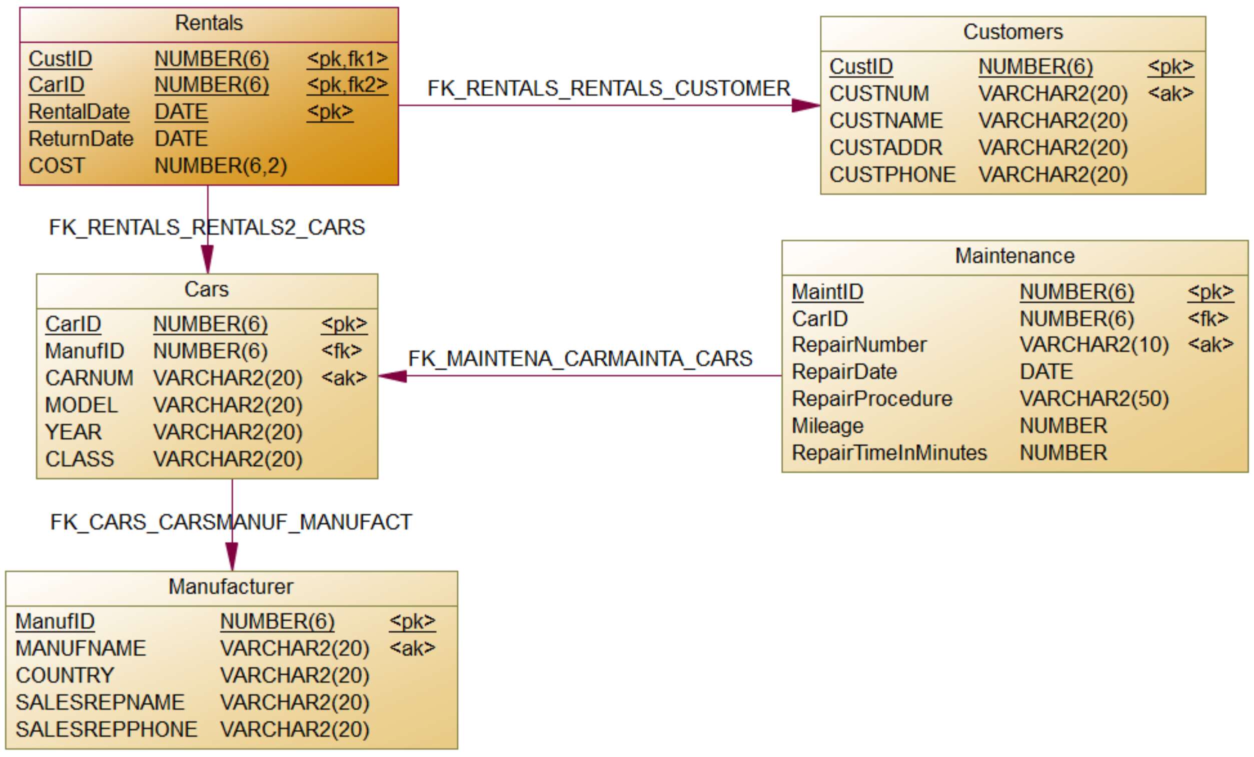


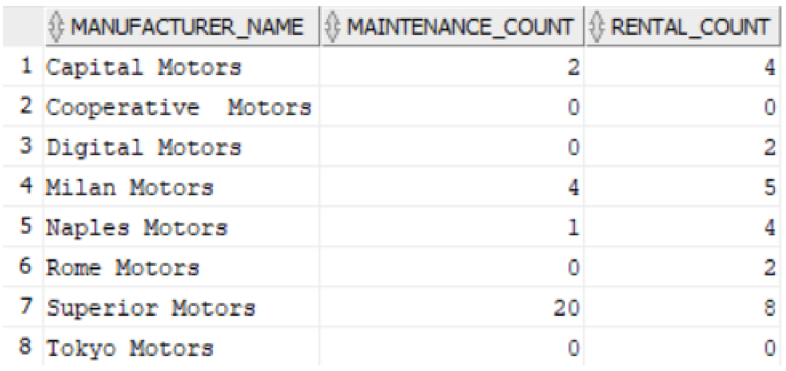
Figure 2. Physical Model

3 SQL Queries

3.1 Query #1

Create materialized view with stores name of manufacturer, count of all repairs of all cars of current manufacturer, count of all car rentals for current manufacturer.
If manufacturer doesn't have rentals or repairs this anyway should be included in report.

```
CREATE MATERIALIZED VIEW MANUFACTURER_SUMMARY AS
SELECT M.MANUFNAME AS MANUFACTURER_NAME,
     COALESCE(COUNT(DISTINCT R.MAINTID), 0) AS MAINTENANCE_COUNT,
     COALESCE(COUNT(DISTINCT C.CARID), 0) AS RENTAL_COUNT
FROM MANUFACTURER M
LEFT JOIN CARS C ON M.MANUFID = C.MANUFID
LEFT JOIN MAINTENANCE R ON C.CARID = R.CARID
GROUP BY M.MANUFID, M.MANUFNAME;
```

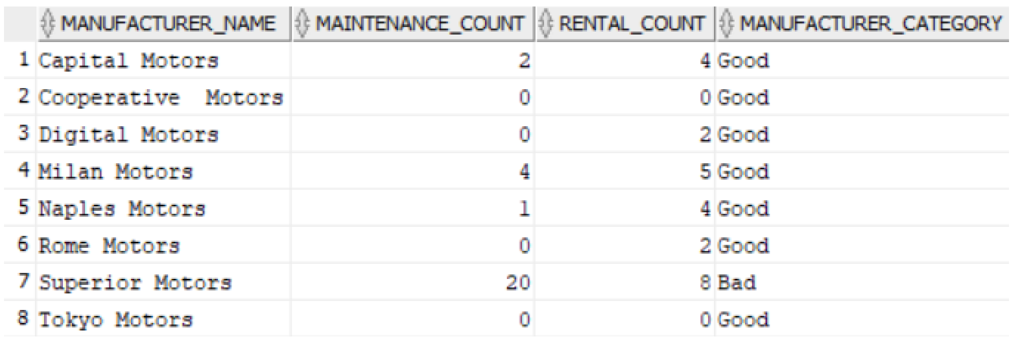


	MANUFACTURER_NAME	MAINTENANCE_COUNT	RENTAL_COUNT
1	Capital Motors	2	4
2	Cooperative Motors	0	0
3	Digital Motors	0	2
4	Milan Motors	4	5
5	Naples Motors	1	4
6	Rome Motors	0	2
7	Superior Motors	20	8
8	Tokyo Motors	0	0

3.2 Query #2

Using materialized view created in Query 1. Create query that extract all data from materialized view and adds category of manufacturer (additional attribute). If manufacturer maintenances are more than rentals then manufacturer is bad, in other case – good.

```
SELECT MS.MANUFACTURER_NAME,
     MS.MAINTENANCE_COUNT,
     MS.RENTAL_COUNT,
     CASE WHEN MS.MAINTENANCE_COUNT > MS.RENTAL_COUNT THEN 'Bad'
         ELSE 'Good'
     END AS MANUFACTURER_CATEGORY
FROM MANUFACTURER_SUMMARY MS;
```

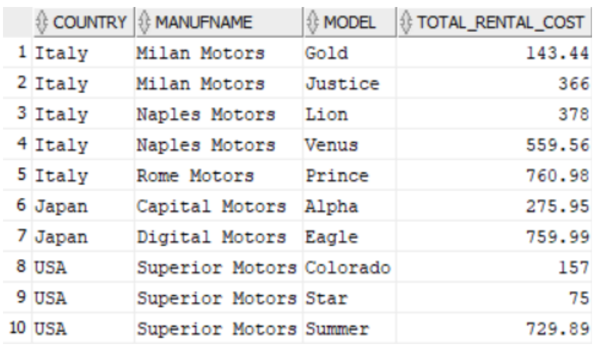


	MANUFACTURER_NAME	MAINTENANCE_COUNT	RENTAL_COUNT	MANUFACTURER_CATEGORY
1	Capital Motors	2	4	Good
2	Cooperative Motors	0	0	Good
3	Digital Motors	0	2	Good
4	Milan Motors	4	5	Good
5	Naples Motors	1	4	Good
6	Rome Motors	0	2	Good
7	Superior Motors	20	8	Bad
8	Tokyo Motors	0	0	Good

3.3 Query #3

You need to create dimension with the following levels: Country, Manufacturer Name, Car Model; and calculate summary of rental cost for every level of dimension including grand total from all levels.
Sort results by country, then by manufacturer name, then by car model.

```
SELECT M.COUNTRY,
    M.MANUFNAME,
    C.MODEL,
    SUM(R.COST) AS TOTAL_RENTAL_COST
FROM RENTALS R
JOIN CARS C ON R.CARID = C.CARID
JOIN MANUFACTURER M ON C.MANUFID = M.MANUFID
GROUP BY M.COUNTRY, M.MANUFNAME, C.MODEL
ORDER BY M.COUNTRY, M.MANUFNAME, C.MODEL;
```

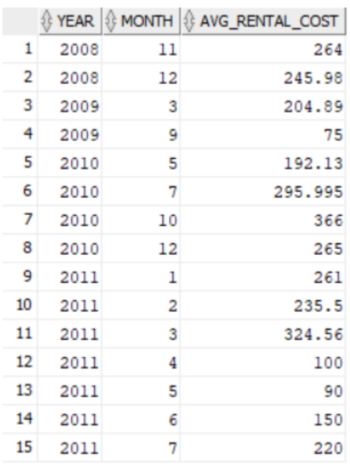


	COUNTRY	MANUFNAME	MODEL	TOTAL_RENTAL_COST
1	Italy	Milan Motors	Gold	143.44
2	Italy	Milan Motors	Justice	366
3	Italy	Naples Motors	Lion	378
4	Italy	Naples Motors	Venus	559.56
5	Italy	Rome Motors	Prince	760.98
6	Japan	Capital Motors	Alpha	275.95
7	Japan	Digital Motors	Eagle	759.99
8	USA	Superior Motors	Colorado	157
9	USA	Superior Motors	Star	75
10	USA	Superior Motors	Summer	729.89

3.4 Query #4

Calculate average values of rentals cost in every month, every year and average across all periods, in one query.
Sort results by year, then by month.

```
SELECT EXTRACT(YEAR FROM RENTALDATE) AS YEAR,
    EXTRACT(MONTH FROM RENTALDATE) AS MONTH,
    AVG(COST) AS AVG_RENTAL_COST
FROM RENTALS
GROUP BY EXTRACT(YEAR FROM RENTALDATE), EXTRACT(MONTH FROM
RENTALDATE)
ORDER BY YEAR, MONTH;
```

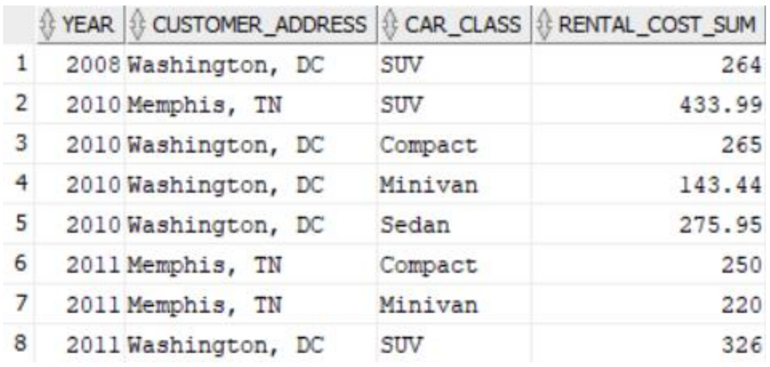


	YEAR	MONTH	AVG_RENTAL_COST
1	2008	11	264
2	2008	12	245.98
3	2009	3	204.89
4	2009	9	75
5	2010	5	192.13
6	2010	7	295.995
7	2010	10	366
8	2010	12	265
9	2011	1	261
10	2011	2	235.5
11	2011	3	324.56
12	2011	4	100
13	2011	5	90
14	2011	6	150
15	2011	7	220

3.5 Query #5

Create OLAP cube with the following dimensions: year of rental, customer address, car class.
Summary of rental cost is fact attribute.
OLAP cube should include data only from 2010 and 2011 years and only for customers who address has letter 't' in customer's address.
Sort data by year, address, class.

```
SELECT
   EXTRACT(YEAR FROM R.RENTALDATE) AS YEAR,
   C.CUSTADDR AS CUSTOMER_ADDRESS,
   CA.CLASS AS CAR_CLASS,
   SUM(R.COST) AS RENTAL_COST_SUM
FROM
   RENTALS R
   JOIN CUSTOMERS C ON R.CUSTID = C.CUSTID
   JOIN CARS CA ON R.CARID = CA.CARID
WHERE
   EXTRACT(YEAR FROM R.RENTALDATE) IN (2010, 2011)
   AND C.CUSTADDR LIKE '%T%'
   OR C.CUSTADDR LIKE '%t%'
GROUP BY
   EXTRACT(YEAR FROM R.RENTALDATE),
   C.CUSTADDR,
   CA.CLASS
ORDER BY
   YEAR, CUSTOMER_ADDRESS, CAR_CLASS;
```



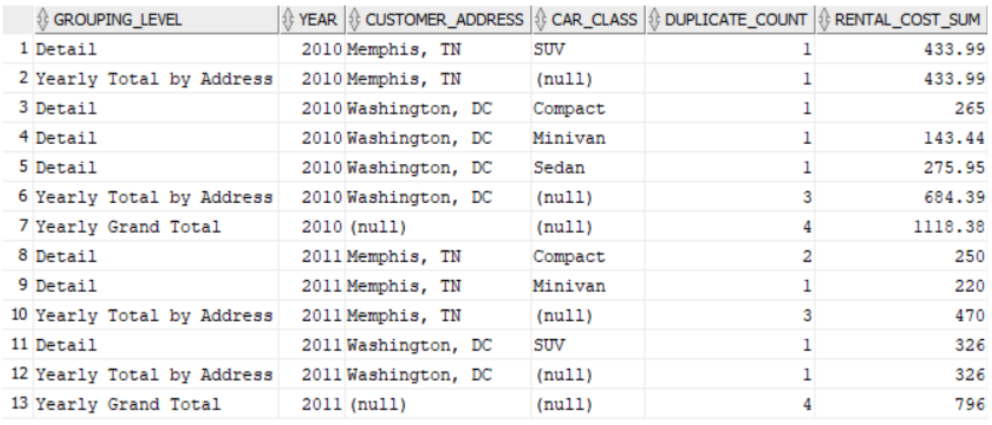
	YEAR	CUSTOMER_ADDRESS	CAR_CLASS	RENTAL_COST_SUM
1	2008	Washington, DC	SUV	264
2	2010	Memphis, TN	SUV	433.99
3	2010	Washington, DC	Compact	265
4	2010	Washington, DC	Minivan	143.44
5	2010	Washington, DC	Sedan	275.95
6	2011	Memphis, TN	Compact	250
7	2011	Memphis, TN	Minivan	220
8	2011	Washington, DC	SUV	326

3.6 Query #6

With OLAP cube made in Q5:

- modify to see only:
    - totals for each year, each address and each class.
    - grand total from all dimensions.
- add grouping columns for every attribute to see if NULL is generated by DBMS;
- show grouping level;
- check if there is duplicated data;

```
SELECT
   CASE
      WHEN EXTRACT(YEAR FROM R.RENTALDATE) IS NOT NULL
          AND C.CUSTADDR IS NOT NULL
          AND CA.CLASS IS NOT NULL THEN 'Detail'
      WHEN EXTRACT(YEAR FROM R.RENTALDATE) IS NOT NULL
          AND C.CUSTADDR IS NOT NULL
          AND CA.CLASS IS NULL THEN 'Yearly Total by Address'
      WHEN EXTRACT(YEAR FROM R.RENTALDATE) IS NOT NULL
          AND C.CUSTADDR IS NULL
          AND CA.CLASS IS NULL THEN 'Yearly Grand Total'
      ELSE 'Grand Total'
   END AS GROUPING_LEVEL,
   EXTRACT(YEAR FROM R.RENTALDATE) AS YEAR,
   C.CUSTADDR AS CUSTOMER_ADDRESS,
   CA.CLASS AS CAR_CLASS,
   COUNT(*) AS DUPLICATE_COUNT,
   SUM(R.COST) AS RENTAL_COST_SUM
FROM
   RENTALS R
   JOIN CUSTOMERS C ON R.CUSTID = C.CUSTID
   JOIN CARS CA ON R.CARID = CA.CARID
WHERE
   EXTRACT(YEAR FROM R.RENTALDATE) IN (2010, 2011)
   AND (C.CUSTADDR LIKE '%T%' OR C.CUSTADDR LIKE '%t%')
GROUP BY
   GROUPING SETS (
      (EXTRACT(YEAR FROM R.RENTALDATE), C.CUSTADDR, CA.CLASS),
      (EXTRACT(YEAR FROM R.RENTALDATE), C.CUSTADDR),
      (EXTRACT(YEAR FROM R.RENTALDATE))
   )
HAVING
   EXTRACT(YEAR FROM R.RENTALDATE) IS NOT NULL
   OR (EXTRACT(YEAR FROM R.RENTALDATE) IS NULL
      AND C.CUSTADDR IS NULL
      AND CA.CLASS IS NULL)
ORDER BY
   YEAR, CUSTOMER_ADDRESS, CAR_CLASS;
```

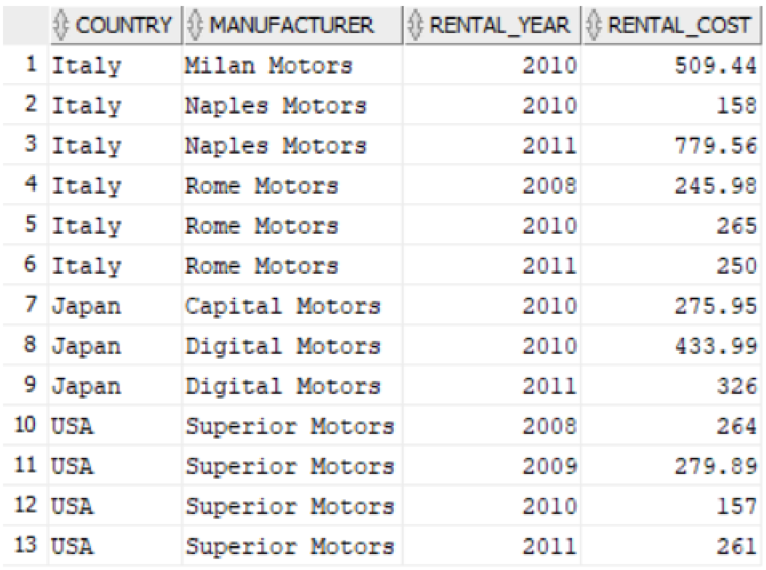


	GROUPING_LEVEL	YEAR	CUSTOMER_ADDRESS	CAR_CLASS	DUPLICATE_COUNT	RENTAL_COST_SUM
1	Detail	2010	Memphis, TN	SUV	1	433.99
2	Yearly Total by Address	2010	Memphis, TN	(null)	1	433.99
3	Detail	2010	Washington, DC	Compact	1	265
4	Detail	2010	Washington, DC	Minivan	1	143.44
5	Detail	2010	Washington, DC	Sedan	1	275.95
6	Yearly Total by Address	2010	Washington, DC	(null)	3	684.39
7	Yearly Grand Total	2010	(null)	(null)	4	1118.38
8	Detail	2011	Memphis, TN	Compact	2	250
9	Detail	2011	Memphis, TN	Minivan	1	220
10	Yearly Total by Address	2011	Memphis, TN	(null)	3	470
11	Detail	2011	Washington, DC	SUV	1	326
12	Yearly Total by Address	2011	Washington, DC	(null)	1	326
13	Yearly Grand Total	2011	(null)	(null)	4	796

3.7 *Query #7*

Create materialized view which stores countries, name of manufacturer, rental year from rental date, summarize rentals costs for particular record.

```
CREATE MATERIALIZED VIEW RENTAL_SUMMARY_MV
BUILD IMMEDIATE
REFRESH COMPLETE
AS
SELECT
   M.COUNTRY AS COUNTRY,
   M.MANUFNAME AS MANUFACTURER,
   EXTRACT(YEAR FROM R.RENTALDATE) AS RENTAL_YEAR,
   SUM(R.COST) AS RENTAL_COST
FROM
   RENTALS R
   JOIN CARS C ON R.CARID = C.CARID
   JOIN MANUFACTURER M ON C.MANUFID = M.MANUFID
GROUP BY
   M.COUNTRY,
   M.MANUFNAME,
   EXTRACT(YEAR FROM R.RENTALDATE)
ORDER BY
   COUNTRY, MANUFACTURER, RENTAL_YEAR;
```



	COUNTRY	MANUFACTURER	RENTAL_YEAR	RENTAL_COST
1	Italy	Milan Motors	2010	509.44
2	Italy	Naples Motors	2010	158
3	Italy	Naples Motors	2011	779.56
4	Italy	Rome Motors	2008	245.98
5	Italy	Rome Motors	2010	265
6	Italy	Rome Motors	2011	250
7	Japan	Capital Motors	2010	275.95
8	Japan	Digital Motors	2010	433.99
9	Japan	Digital Motors	2011	326
10	USA	Superior Motors	2008	264
11	USA	Superior Motors	2009	279.89
12	USA	Superior Motors	2010	157
13	USA	Superior Motors	2011	261

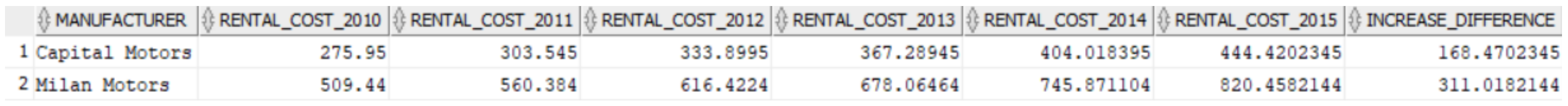
3.8 Query #8

Using materialized view created in Q7:
Create forecast of planned income from rental costs for manufacturers Capital Motors and Milan Motors (result for every manufacturer should appear):

- calculate planned rentals costs from 2011 till 2015 using existing income from rental costs in 2010, assuming that every year income will increase by 10%.

Add attribute that calculates difference between current year and previous year.
Sort data by manufacturer, year.

```
SELECT
   MANUFACTURER,
   RENTAL_COST AS RENTAL_COST_2010,
   RENTAL_COST * 1.1 AS RENTAL_COST_2011,
   RENTAL_COST * 1.1 * 1.1 AS RENTAL_COST_2012,
   RENTAL_COST * 1.1 * 1.1 * 1.1 AS RENTAL_COST_2013,
   RENTAL_COST * 1.1 * 1.1 * 1.1 * 1.1 AS RENTAL_COST_2014,
   RENTAL_COST * 1.1 * 1.1 * 1.1 * 1.1 * 1.1 AS RENTAL_COST_2015,
   RENTAL_COST * 1.1 * 1.1 * 1.1 * 1.1 * 1.1 - RENTAL_COST AS
INCREASE_DIFFERENCE
FROM
   rental_summary_mv
WHERE
   MANUFACTURER IN ('Capital Motors', 'Milan Motors')
   AND RENTAL_YEAR = 2010
GROUP BY
   MANUFACTURER,
   RENTAL_COST
ORDER BY
   MANUFACTURER;
```



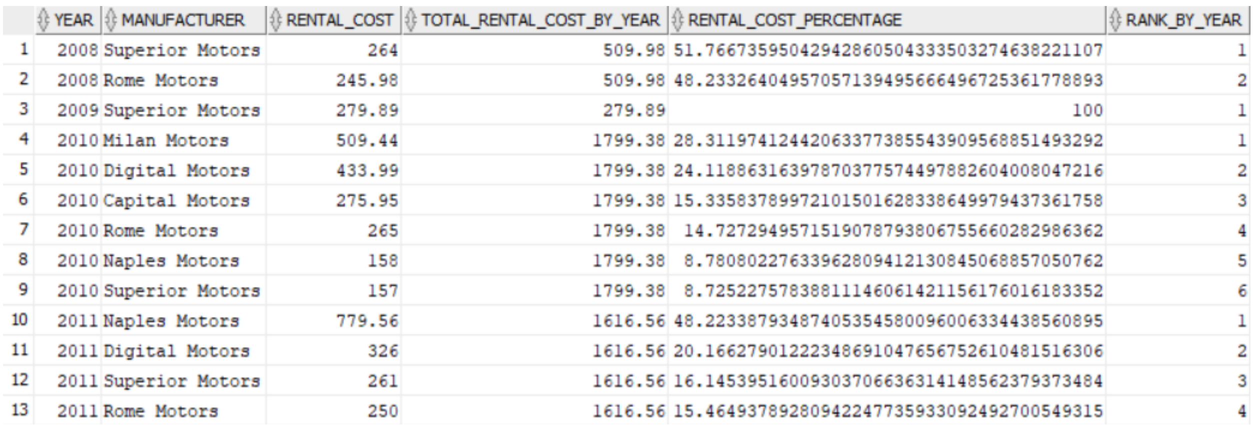
	MANUFACTURER	RENTAL_COST_2010	RENTAL_COST_2011	RENTAL_COST_2012	RENTAL_COST_2013	RENTAL_COST_2014	RENTAL_COST_2015	INCREASE_DIFFERENCE
1	Capital Motors	275.95	303.545	333.8995	367.28945	404.018395	444.4202345	168.4702345
2	Milan Motors	509.44	560.384	616.4224	678.06464	745.871104	820.4582144	311.0182144

3.9 Query #9

Using materialized view created in Q7:

- extract year, manufacturers name, rental total cost;
- add an attribute that shows sum of rental costs in particular (partitioned by) year;
- add an attribute that calculates percentage of current manufacturer rental costs divided by total rental costs in particular (partitioned by) year;
- add an attribute that calculates rank for manufacturers inside particular (partitioned by) year based on rental cost.

```
SELECT
   RENTAL_YEAR AS YEAR,
   MANUFACTURER,
   RENTAL_COST,
   SUM(RENTAL_COST) OVER (PARTITION BY RENTAL_YEAR) AS
TOTAL_RENTAL_COST_BY_YEAR,
   RENTAL_COST / SUM(RENTAL_COST) OVER (PARTITION BY RENTAL_YEAR) *
100 AS RENTAL_COST_PERCENTAGE,
   RANK() OVER (PARTITION BY RENTAL_YEAR ORDER BY RENTAL_COST DESC)
AS RANK_BY_YEAR
FROM
   RENTAL_SUMMARY_MV
ORDER BY
   RENTAL_YEAR,
   RENTAL_COST DESC;
```



	YEAR	MANUFACTURER	RENTAL_COST	TOTAL_RENTAL_COST_BY_YEAR	RENTAL_COST_PERCENTAGE	RANK_BY_YEAR
1	2008	Superior Motors	264	509.98	51.766735950429428605043335032746382211O7	1
2	2008	Rome Motors	245.98	509.98	48.2332640495705713949566649672536177889З	2
3	2009	Superior Motors	279.89	279.89	100	1
4	2010	Milan Motors	509.44	1799.38	28.31197412442063377385543909568851493292	1
5	2010	Digital Motors	433.99	1799.38	24.11886316397870377574497882604008047216	2
6	2010	Capital Motors	275.95	1799.38	15.33583789972101501628338649979437361758	3
7	2010	Rome Motors	265	1799.38	14.72729495715190787938067556602829863 62	4
8	2010	Naples Motors	158	1799.38	8.780802276339628094121308450688570507 62	5
9	2010	Superior Motors	157	1799.38	8.725227578388111460614211561760161833 52	6
10	2011	Naples Motors	779.56	1616.56	48.22338793487405354580096006334438560895	1
11	2011	Digital Motors	326	1616.56	20.16627901222348691047656752610481516306	2
12	2011	Superior Motors	261	1616.56	16.14539516009303706636314148562379373484	3
13	2011	Rome Motors	250	1616.56	15.46493789280942247735933092492700549315	4

3.10 Query #10

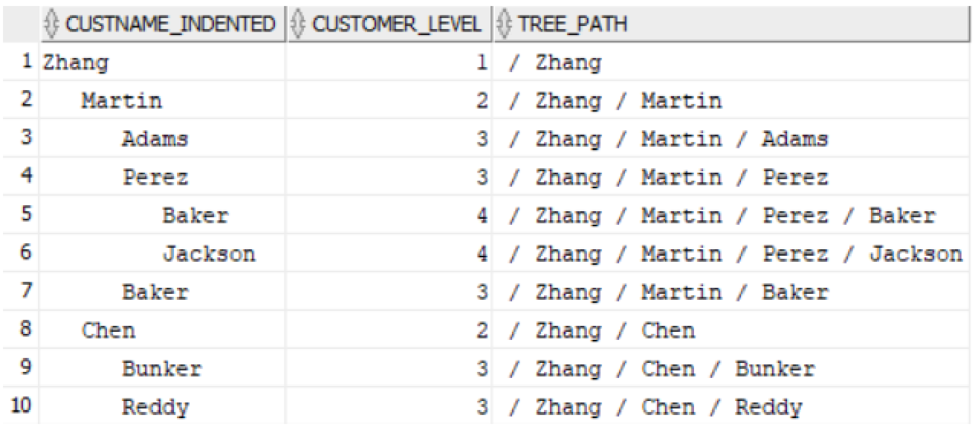
Extract from customers:

- names adding indent of 3 "spaces" for every next level;
- show customer level number in the hierarchy;
- show tree path from current customer up to top customer in the hierarchy;
- show only first 4 levels of the tree.

```
ALTER TABLE CUSTOMERS ADD REFCUST NUMBER(6);
ALTER TABLE CUSTOMERS
   ADD CONSTRAINT CUSTREFHIER
      FOREIGN KEY (REFCUST)
         REFERENCES CUSTOMERS(CUSTID);

UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Zhang') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '133819');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Zhang') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '182194');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Martin') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '246754');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Martin') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '285028');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Martin') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '294827');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Chen') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '381074');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Chen') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '429292');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Perez') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '482910');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Perez') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '539118');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Jackson') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '592937');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Jackson') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '730282');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Jefferson') WHERE CUSTID
= (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '876586');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Jefferson') WHERE CUSTID
= (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '925820');
UPDATE CUSTOMERS SET REFCUST = (SELECT CUSTID FROM CUSTOMERS WHERE CUSTNAME = 'Stewart') WHERE CUSTID =
(SELECT CUSTID FROM CUSTOMERS WHERE CUSTNUM = '956732');
```

```
SELECT LPAD(' ', (LEVEL-1)*3) || CUSTNAME AS CUSTNAME_INDENTED,
    LEVEL AS CUSTOMER_LEVEL,
    SYS_CONNECT_BY_PATH(CUSTNAME, ' / ') AS TREE_PATH
FROM CUSTOMERS
START WITH REFCUST IS NULL
CONNECT BY PRIOR CUSTID = REFCUST
AND LEVEL <= 4;
```



```
    CUSTNAME_INDENTED   CUSTOMER_LEVEL  TREE_PATH
 1  Zhang                            1  / Zhang
 2     Martin                        2  / Zhang / Martin
 3        Adams                      3  / Zhang / Martin / Adams
 4        Perez                      3  / Zhang / Martin / Perez
 5           Baker                   4  / Zhang / Martin / Perez / Baker
 6           Jackson                 4  / Zhang / Martin / Perez / Jackson
 7        Baker                      3  / Zhang / Martin / Baker
 8     Chen                          2  / Zhang / Chen
 9        Bunker                     3  / Zhang / Chen / Bunker
10        Reddy                      3  / Zhang / Chen / Reddy
```