

Database Management Systems Project

Garrett Kilcer

- a. Create a query to display unique combination of cities and states from Customers table

```
select distinct city, region  
from customer_f  
order by city;
```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, 'File', 'Edit', 'View', 'Designer', 'Run Source', 'Tools', 'Windows', and 'Help' are visible. A connection named 'myconnection' is selected. The left sidebar contains a tree view of database objects: Connections, Oracle Connectors, Tables, Views, Indexes, Packages, Procedures, Functions, Operators, Sequences, Types, Sequences, Memoryviews, Materialized V, Synonyms, Public synonym, Database Link, Public Database, Directories, Job Schedules, and XML Schema. Below the sidebar is the 'Worksheet' tab, which is active and titled 'Query Builder'. The worksheet pane displays the following SQL query:

```
select distinct city, region  
from customer_f  
order by city;
```

Below the worksheet is the 'Query Result' pane, which shows the output of the query:

CITY	REGION
Atlanta	SOUT
New York	SOUT
Washington	SOUT

- b. Create a query to display the last name concatenated with the city, separated by space.

Name this column CUSTOMER_AND_CITY

```
select concat (last_name, concat(' ', city)) as CUSTOMER_AND_CITY  
from customer_f;
```

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active and titled 'Query Builder'. The worksheet pane displays the following SQL query:

```
select concat (last_name, concat(' ', city)) as CUSTOMER_AND_CITY  
from customer_f;
```

Below the worksheet is the 'Query Result' pane, which shows the output of the query:

CUSTOMER_AND_CITY
ANTONETTE Atlanta
ZORIEEN New York
MONI Washington

- c. Create a query to display first name, last name, join date, monthly discount, monthly discount after an addition of 20% and monthly discount after a reduction of 20%.

```
select first_name, last_name, join_date, monthly_discount,  
monthly_discount+(monthly_discount *.20),  
monthly_discount-(monthly_discount*.20) from acdb_customers;
```

Database Management Systems Project

```
select first_name, last_name, join_date, monthly_discount, monthly_discount+(monthly_discount *.20), monthly_discount-(monthly_discount*.20) from acdb_customers;
```

Query Result:

	FIRST_NAME	LAST_NAME	JOIN_DATE	MONTHLY_DISCOUNT	MONTHLY_DISCOUNT+(MONTHLY_DISCOUNT*.20)	MONTHLY_DISCOUNT-(MONTHLY_DISCOUNT*.20)
1	Cedric	Clarke	11-SEP-05	5	5.4	5.6
2	Clark	James	17-MAR-05	28	33.4	22.4
3	Cortes	Morgan	11-DEC-03	13	15.4	10.4
4	Kelly	Ruthes	01-FEB-07	9	10.8	7.2
5	Lake	Edwards	08-JUL-07	(null)	(null)	(null)
6	Delos	Hill	19-APR-09	15	18	12
7	Roland	Moore	15-JUN-08	25	30	20
8	Mike	Clark	05-JAN-09	3	3.6	2.4
9	Emery	Harrison	20-SEP-09	5	11.8	7.2
10	Leandro	Scott	07-OCT-09	9	11.8	7.2
11	Thomas	Young	01-OCT-09	(null)	(null)	(null)
12	Dominic	Morris	28-NAU-08	18	18	12
13	Tyler	Ball	03-AUG-08	20	24	16

- d. All customers who live in New York **and** whose monthly discount is in the range between 30 and 40

select * from acdb_customers where city='New York' and monthly_discount between '30' and '40';

```
select * from acdb_customers where city='New York' and monthly_discount between '30' and '40';
```

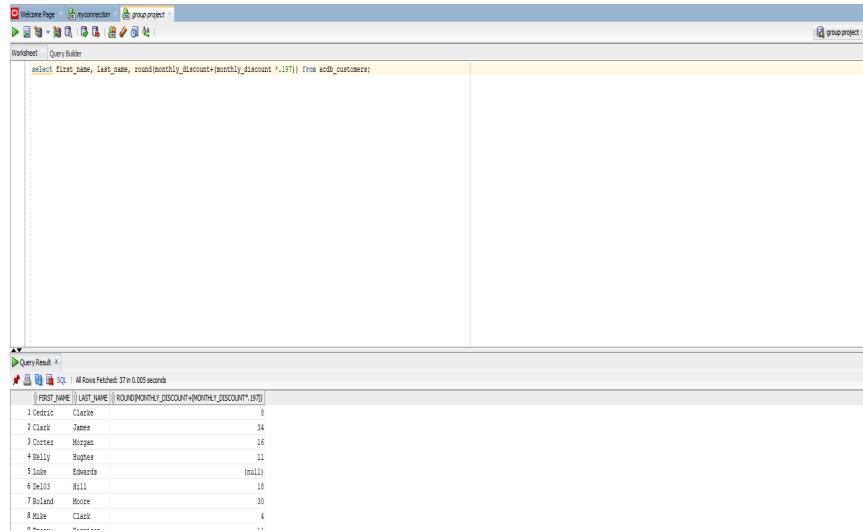
Query Result:

CUSTOMER_ID	FIRST_NAME	LAST_NAME	CITY	STATE	STREET	MAIN_PHONE	SECOND_PHONE	FAX	MONTHLY_DISCOUNT	PACK_ID	BIRTH_DATE	JOIN_DATE
1	Cedric	Clarke	New York	NY	123 Main St	555-1234	555-1234	555-1234	10.4	1	1970-01-01	2005-09-11
2	Clark	James	New York	NY	456 Elm St	555-2345	555-2345	555-2345	22.4	2	1970-03-17	2005-05-17
3	Cortes	Morgan	New York	NY	789 Oak St	555-3456	555-3456	555-3456	15.4	3	1970-12-11	2003-12-11
4	Kelly	Ruthes	New York	NY	123 Maple St	555-4567	555-4567	555-4567	7.2	4	1970-02-01	2007-01-07
5	Lake	Edwards	New York	NY	456 Pine St	555-5678	555-5678	555-5678	11.8	5	1970-07-08	2009-06-08
6	Delos	Hill	New York	NY	789 Birch St	555-6789	555-6789	555-6789	18	6	1970-04-19	2009-04-19
7	Roland	Moore	New York	NY	123 Cedar St	555-7890	555-7890	555-7890	30	7	1970-06-15	2008-06-15
8	Mike	Clark	New York	NY	456 Cypress St	555-8901	555-8901	555-8901	2.4	8	1970-01-05	2009-01-05
9	Emery	Harrison	New York	NY	789 Fir St	555-9802	555-9802	555-9802	15.4	9	1970-09-20	2009-09-20
10	Leandro	Scott	New York	NY	123 Birch St	555-0913	555-0913	555-0913	11.8	10	1970-07-07	2009-07-07
11	Thomas	Young	New York	NY	456 Cedar St	555-1014	555-1014	555-1014	(null)	11	1970-01-01	2009-01-01
12	Dominic	Morris	New York	NY	789 Fir St	555-2015	555-2015	555-2015	18	12	1970-08-28	2008-08-28
13	Tyler	Ball	New York	NY	123 Fir St	555-3016	555-3016	555-3016	24	13	1970-08-03	2008-08-03

- e. From customers table, display monthly discount after addition of 19.7%, expressed as a whole number

select first_name, last_name, round(monthly_discount+(monthly_discount *.197)) from acdb_customers;

Database Management Systems Project



Worksheet - Query Builder

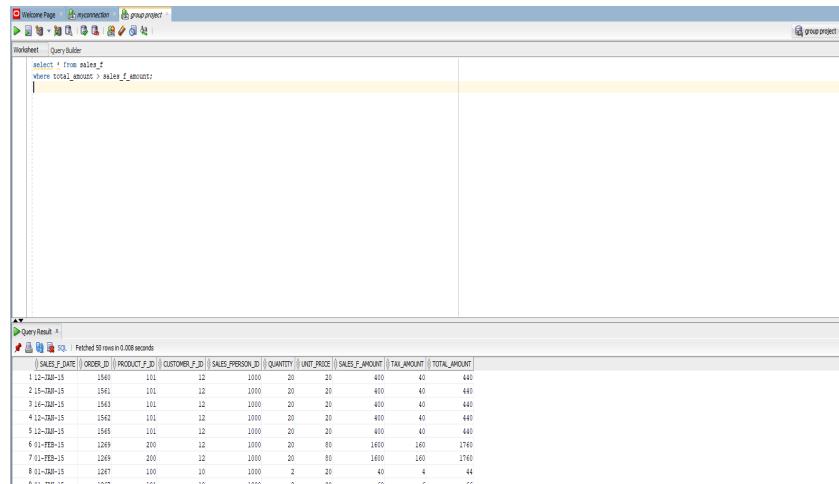
```
select first_name, last_name, round(monthly_discount+monthly_discount*.15) from acdb_customers;
```

Query Result

FIRST_NAME	LAST_NAME	ROUND(MONTHLY_DISCOUNT+MONTHLY_DISCOUNT*.15)
Cedric	Claire	8
Clark	Jane	14
Cortez	Hughes	16
DeWitt	Boggs	11
Diane	Edwards	(null)
DeWitt	Hill	18
Roland	Horne	30
Mike	Clark	4
...

- f. Produce a list of rows showing all columns from the SALES table where total amount is greater than sales amount.

```
select * from sales_f
where total_amount > sales_f_amount;
```



Worksheet - Query Builder

```
select * from sales_f
where total_amount > sales_f_amount;
```

Query Result

SALES_F_DATE	ORDER_ID	PRODUCT_F_ID	CUSTOMER_F_ID	SALES_PERSON_F_ID	QUANTITY	UNIT_PRICE	SALES_F_AMOUNT	TAX_AMOUNT	TOTAL_AMOUNT
11-3-15	1560	101	12	1000	20	20	410	40	440
21-3-15	1561	101	12	1000	20	20	410	40	440
31-3-15	1563	101	12	1000	20	20	410	40	440
41-3-15	1564	101	12	1000	20	20	410	40	440
51-3-15	1565	101	12	1000	20	20	410	40	440
61-3-15	1249	200	12	1000	20	80	1610	160	1770
71-3-15	1249	200	12	1000	20	80	1610	160	1770
81-3-15	1247	100	10	1000	2	20	40	4	44
91-3-15	1247	101	10	1000	2	20	40	4	44

- g. Produce a list of rows showing sales date, product number, order number, sales amount, tax amount and a calculated column (sum of sales amount and tax amount) from the SALES table

```
select sales_f_date, product_f_id, order_id, sales_f_amount, tax_amount, tax_amount
+ sales_f_amount
from sales_f;
```

Database Management Systems Project

```

select sales_f_date, product_f_id, order_id, sales_f_amount, tax_amount, tax_amount + sales_f_amount
from sales_f;

```

Query Result:

SALES_F_DATE	PRODUCT_F_ID	ORDER_ID	SALES_F_AMOUNT	TAX_AMOUNT	TAX_AMOUNT+SALES_F_AMOUNT
1.12-2015	101	1540	400	40	440
2.15-2015	101	1541	400	40	440
3.14-2015	101	1542	400	40	440
4.12-2015	101	1543	400	40	440
5.12-2015	101	1545	400	40	440
6.01-2015	200	1249	1600	160	1760
7.01-2015	200	1249	1600	160	1760
8.01-2015	100	1247	40	4	44
9.01-2015	101	1247	60	6	66
10.02-2015	100	1248	300	30	330
11.02-2015	100	1249	1600	160	1760
12.04-2015	104	1270	500	50	550
13.10-2015	101	1271	300	30	330

- h. Produce a list of rows showing all columns from the SALES table where quantity plus 10 is greater than or equal to 20

```
select * from sales_f where quantity + 10 >= 20;
```

```

select * from sales_f;
select * from sales_f where quantity + 10 >= 20;

```

Query Result:

SALES_F_DATE	PRODUCT_F_ID	ORDER_ID	SALES_F_AMOUNT	TAX_AMOUNT	TOTAL_AMT
1.12-2015	100	1240	1000	20	1020
1.12-2015	101	1241	1000	20	1020
1.12-2015	101	1242	1000	20	1020
1.12-2015	101	1243	1000	20	1020
1.12-2015	101	1245	1000	20	1020
6.01-2015	200	1249	1000	20	1020
7.01-2015	200	1249	1000	20	1020
8.01-2015	100	1247	100	20	120
9.01-2015	101	1247	120	20	140
10.02-2015	100	1248	300	30	330
11.02-2015	100	1249	1000	20	1020
12.04-2015	104	1270	500	50	550
13.10-2015	101	1271	300	30	330

- i. Produce a list of rows showing sales date, product number, order number, sales amount, tax amount from the SALES table and sort it by column sales amount and tax amount in ascending order

```
select sales_f_date, product_f_id, order_id, sales_f_amount, tax_amount from sales_f
order by sales_f_amount, tax_amount asc;
```

Database Management Systems Project

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, 'File', 'Edit', 'View', 'Navigate', 'Run', 'Source', 'Tools', 'Window', and 'Help' are visible. Below the menu is a toolbar with icons for new connection, open connection, save, print, and help. The left sidebar has sections for 'Connections' (including 'Oracle Connections' with 'sales_10_12', 'sales_10_14', 'sales_10_19', 'DatabaseName 3', 'HR schema', 'SQL Project', 't1st', 't2nd', 't3rd'), 'Reports' (with 'All Reports', 'Analytic View Reports', 'Data Model Reports', 'Data Model Reports', 'OLAP Reports', 'TimesTen Reports', 'User Defined Reports'), and 'Oracle NoSQL Connections'.

In the main workspace, the 'Welcome Page' for 'SQL_Project' is displayed. The 'SQL Worksheet' tab is active, containing the following SQL code:

```

/* 1. Produce a list of rows showing sales date, product number, order number, sales amount,
   tax amount from the SALES table and sort it by sales amount in ascending order */

select sales_f_date, product_f_id, sales_f_amount, tax_amount from sales_f
order by sales_f_amount, tax_amount asc;

```

The 'Query Result' tab shows the output of the query:

SALES_F_DATE	PRODUCT_F_ID	ORDER_ID	SALES_F_AMOUNT	TAX_AMOUNT
11-12-2015	101	1247	40	4
11-12-2015	101	1247	40	4
11-12-2015	100	1274	140	14
11-12-2015	100	1248	300	30
11-12-2015	101	1275	300	30
11-12-2015	101	1275	300	30
11-12-2015	100	1276	340	34
11-12-2015	101	1264	400	40
11-12-2015	101	1264	400	40
11-12-2015	101	1263	400	40
11-12-2015	101	1262	400	40
11-12-2015	101	1265	400	40

- j. Produce a list of rows showing order number, sales date, product number, sales amount, tax amount from the SALES table and sort it by order number in descending order

```
select order_id, sales_f_date, product_f_id, sales_f_amount, tax_amount from sales_f
order by order_id desc;
```

The screenshot shows the Oracle SQL Developer interface, similar to the previous one. The 'Welcome Page' for 'SQL_Project' is displayed. The 'SQL Worksheet' tab is active, containing the same SQL code as before:

```

/* 1. Produce a list of rows showing sales date, product number, order number, sales amount,
   tax amount from the SALES table and sort it by sales amount and tax amount in ascending order */

select sales_f_date, product_f_id, order_id, sales_f_amount, tax_amount from sales_f
order by sales_f_amount, tax_amount asc;

/* 2. Produce a list of rows showing order number, sales date, product number, sales amount,
   tax amount from the SALES table and sort it by order number in descending order */

select order_id, sales_f_date, product_f_id, sales_f_amount, tax_amount from sales_f
order by order_id desc;

```

The 'Query Result' tab shows the output of the second part of the query:

ORDER_ID	SALES_F_DATE	PRODUCT_F_ID	SALES_F_AMOUNT	TAX_AMOUNT
1	1509 12-2015	101	400	40
2	1509 12-2015	101	400	40
3	1509 14-2015	101	400	40
4	1509 15-2015	101	400	40
5	1509 15-2015	101	400	40
6	1509 12-2015	101	400	40
7	1501 01-MAY-15	100	1140	114
8	1500 30-APR-15	100	2100	210
9	1349 27-APR-15	100	2800	280
10	1349 27-APR-15	100	1140	114
11	1347 27-APR-15	100	1320	132
12	1344 24-APR-15	100	840	84
13	1148 25-APR-15	100	3200	320

- k. Display the first name, birthdate and age for all customers whose older than 50.

```
select first_name, birth_date, round((sysdate - birth_date)/365.25, 0) as age from
acdb_customers where (sysdate - birth_date)/365.25 >= 50;
```

Database Management Systems Project

The screenshot shows the Oracle SQL Developer interface. In the Worksheet window, there is a multi-line text area containing SQL code. In the Query Result window, there is a table with three columns: FIRST_NAME, BIRTH_DATE, and AGE. The data is as follows:

FIRST_NAME	BIRTH_DATE	AGE
2 Luke	04-OCT-66	34
3 Boland	23-JUL-60	40
4 Miles	20-MAR-66	35
5 Sommer	13-MAR-63	32
6 Hayes	05-SEP-63	30
7 Chadelie	07-MAR-63	30
8 Avery	15-JUL-67	26
9 Laddie	02-APR-63	32
10 Kelly	13-NOV-69	22
11 Art	13-NOV-69	22
12 Everett	27-NOV-70	20
13 Alexander	07-JUN-69	22
14 Clark	10-NOV-65	26

- I. Display all the data from Customers table, for all customers whose birthdate is today.

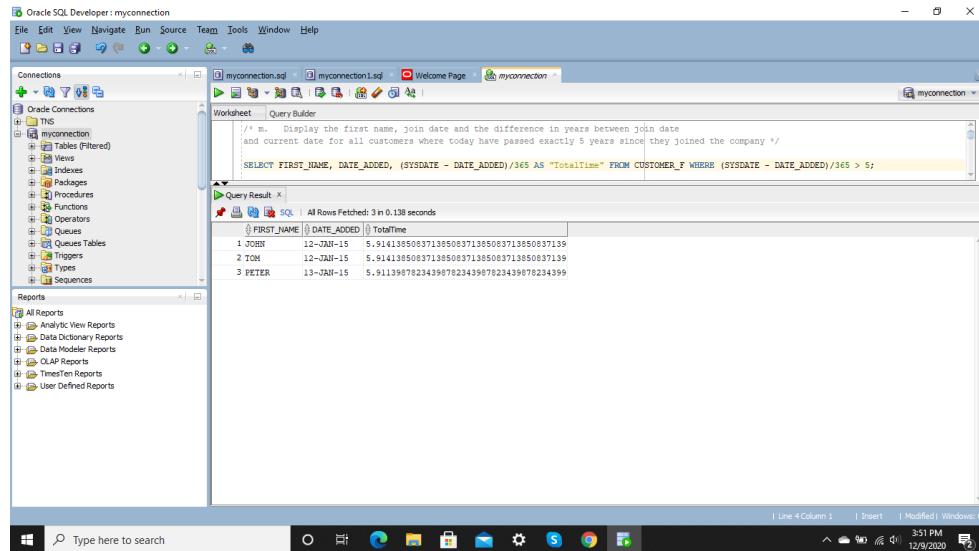
```
select * from acdb_customers where to_char(birth_date, 'DD-MON') =
to_char(sysdate, 'DD-MON');
```

The screenshot shows the Oracle SQL Developer interface. In the Worksheet window, there is a multi-line text area containing SQL code. In the Query Result window, there is a table with six columns: CUSTOMER_ID, FIRST_NAME, LAST_NAME, CITY, STATE, and STREET. The data is as follows:

CUSTOMER_ID	FIRST_NAME	LAST_NAME	CITY	STATE	STREET
1	197 Russell	Campbell	Sunny Valley	California	1444 Wallace Avenue
			537	401	7249
			(mail)		720-537-4523

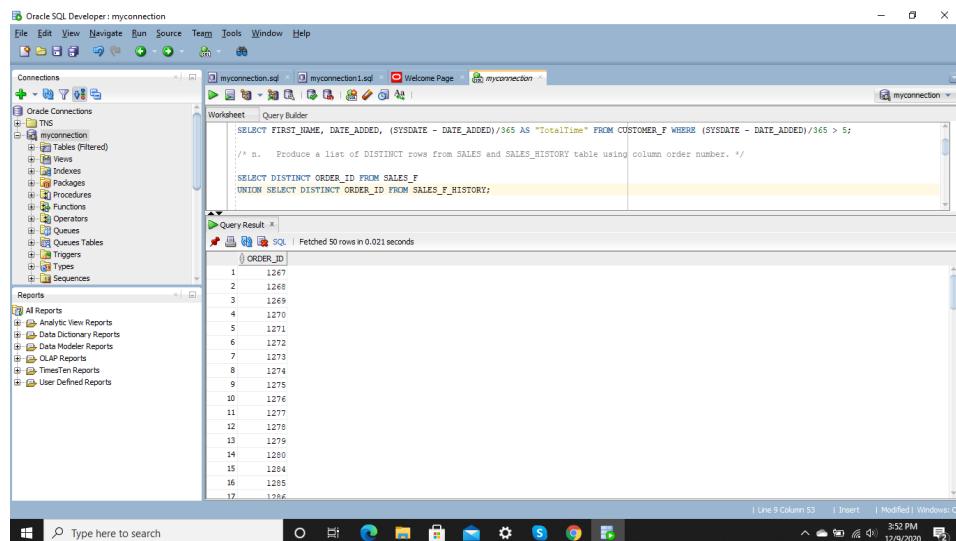
- m. Display the first name, join date and the difference in years between join date and current date for all customers where today have passed exactly 5 years since they joined the company.

Database Management Systems Project



```
SELECT FIRST_NAME, DATE_ADDED, (SYSDATE - DATE_ADDED)/365 AS "TotalTime"  
FROM CUSTOMER_F WHERE (SYSDATE - DATE_ADDED)/365 > 5;
```

- n. Produce a list of DISTINCT rows from SALES and SALES_HISTORY table using column order number.



```
SELECT DISTINCT ORDER_ID FROM SALES_F UNION SELECT DISTINCT ORDER_ID FROM  
SALES_F_HISTORY;
```

- o. Produce a list of rows which are present in SALES table and are not present in SALES_HISTORY table using column order number.
select order_id from sales minus select order_id from sales_history;

Database Management Systems Project

The screenshot shows the Oracle SQL Developer interface. In the central workspace, a query builder window displays the following SQL code:

```
/s 0 /
select order_id from sales minus select order_id from sales_history;
```

The results pane shows a list of 50 ORDER_ID values from 1269 to 1294. The results are as follows:

ORDER_ID
1269
1270
1271
1272
1273
1274
1275
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294

- p. Produce a list which shows maximum, average, minimum sales volume broken by sales date and order number from SALES table

```
select max(sales_amount) over(partition by sales_date, order_id) as "Max",
       avg(sales_amount) over(partition by sales_date, order_id) as "Average",
       min(sales_amount) over(partition by sales_date, order_id) as "Min"
  from sales;
```

The screenshot shows the Oracle SQL Developer interface. In the central workspace, a query builder window displays the following SQL code:

```
/s P /
select max(sales_amount) over(partition by sales_date, order_id) as "Max",
       avg(sales_amount) over(partition by sales_date, order_id) as "Average",
       min(sales_amount) over(partition by sales_date, order_id) as "Min"
  from sales;
```

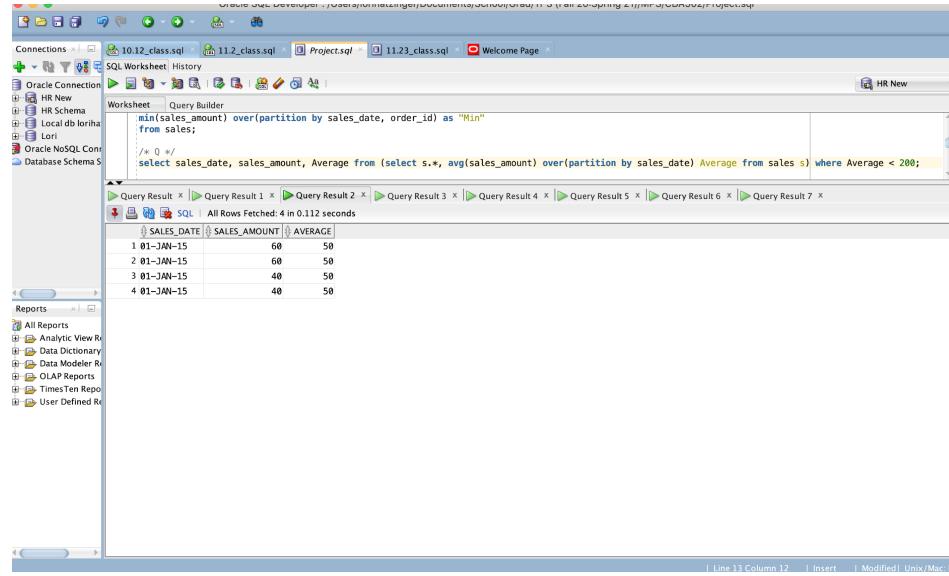
The results pane shows a list of 22 rows, each containing three columns: Max, Average, and Min. The data is as follows:

	Max	Average	Min
1	60	50	40
2	60	50	40
3	60	50	40
4	60	50	40
5	300	300	300
6	300	300	300
7	400	400	400
8	400	400	400
9	400	400	400
10	400	400	400
11	400	400	400
12	400	400	400
13	400	400	400
14	400	400	400
15	400	400	400
16	400	400	400
17	400	400	400
18	400	400	400
19	1600	1600	1600
20	1600	1600	1600
21	1600	1600	1600
22	1600	1600	1600

- q. Produce a list which shows average sales volume broken by sales date from SALES table and average sales volume is less than 200.

```
select sales_date, sales_amount, Average from (select s.*,
           avg(sales_amount) over(partition by sales_date) Average
          from sales s)
 where Average < 200;
```

Database Management Systems Project



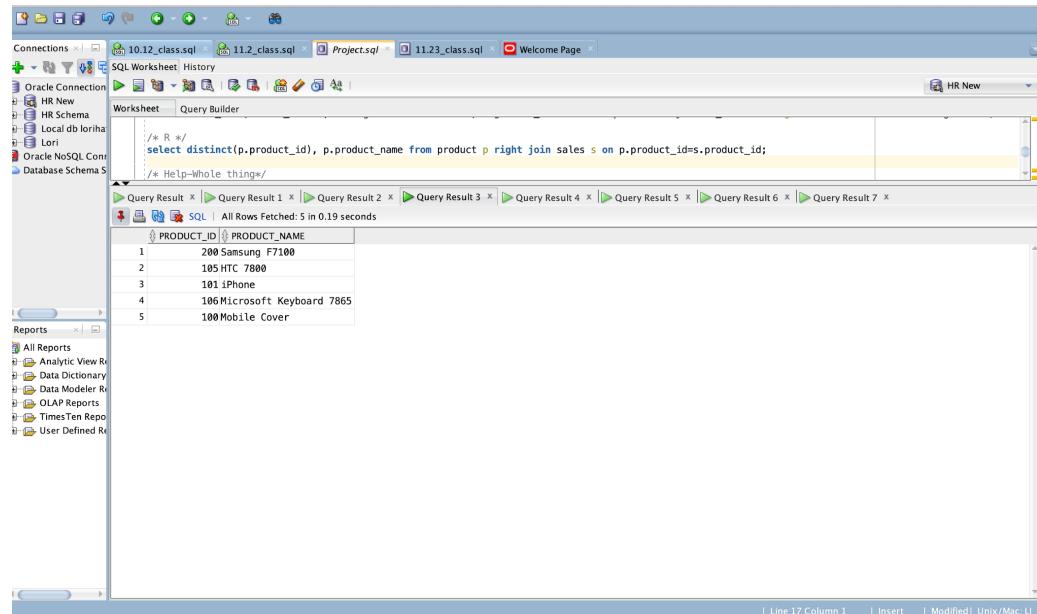
A screenshot of the Oracle SQL Developer interface. The 'Connections' sidebar shows several database connections, including 'HR New'. The 'Worksheet' tab is active, displaying a query builder window with the following SQL code:

```
min(sales_amount) over(partition by sales_date, order_id) as "Min"  
from sales;  
  
/* 0 */  
select sales_date, sales_amount, Average from (select s.*, avg(sales_amount) over(partition by sales_date) Average from sales s) where Average < 200;
```

The results pane shows a table with three columns: SALES_DATE, SALES_AMOUNT, and AVERAGE. The data is as follows:

SALES_DATE	SALES_AMOUNT	AVERAGE
1 01-JAN-15	60	50
2 01-JAN-15	60	50
3 01-JAN-15	40	50
4 01-JAN-15	40	50

- r. Produce a list which returns all products only those products that have had some sales
select distinct(p.product_id), p.product_name from product p right join sales s on p.product_id=s.product_id;



A screenshot of the Oracle SQL Developer interface. The 'Connections' sidebar shows 'HR New' selected. The 'Worksheet' tab is active, displaying a query builder window with the following SQL code:

```
/* R */  
select distinct(p.product_id), p.product_name from product p right join sales s on p.product_id=s.product_id;  
  
/* Help-Whole things/
```

The results pane shows a table with two columns: PRODUCT_ID and PRODUCT_NAME. The data is as follows:

PRODUCT_ID	PRODUCT_NAME
1	200 Samsung F7100
2	105 HTC 7800
3	101 iPhone
4	106 Microsoft Keyboard 7865
5	100 Mobile Cover

- s. Produce a list which returns all products irrespective of whether the product has had any sales
select distinct(product_id), product_name from product;

Database Management Systems Project

The screenshot shows the Oracle SQL Developer interface with a query window open. The query is:

```
/* S */
select distinct(product_id), product_name from product;
```

The results show a list of products:

PRODUCT_ID	PRODUCT_NAME
1	200 Samsung F7108
2	105 HTC 7800
3	101 iPhone
4	106 Microsoft Keyboard 7865
5	501 Microsoft Mouse 7863
6	100 Mobile Cover

- t. Produce a list of rows with all columns from sales table for the sales generated by customer MANN

```
select * from sales s left join customer c on s.customer_id=s.customer_id where c.last_name = 'MANN';
```

The screenshot shows the Oracle SQL Developer interface with a query window open. The query is:

```
/* T */
select * from sales s left join customer c on s.customer_id=s.customer_id where c.last_name = 'MANN';
```

The results show a list of sales rows for customer MANN:

SALES_DATE	ORDER_ID	PRODUCT_ID	CUSTOMER_ID	SALESPERSON_ID	QUANTITY	UNIT_PRICE	SALES_AMOUNT	TAX_AMOUNT	TOTAL_AMOUNT	CUSTOMER_ID_1	FIRST_I
1 12-JAN-15	1560	101	12	1000	20	20	400	40	440	12 PETER	
2 15-JAN-15	1561	101	12	1000	20	20	400	40	440	12 PETER	
3 16-JAN-15	1563	101	12	1000	20	20	400	40	440	12 PETER	
4 12-JAN-15	1562	101	12	1000	20	20	400	40	440	12 PETER	
5 12-JAN-15	1565	101	12	1000	20	20	400	40	440	12 PETER	
6 01-FEB-15	1269	200	12	1000	20	80	1600	160	1760	12 PETER	
7 01-FEB-15	1269	200	12	1000	20	80	1600	160	1760	12 PETER	
8 01-JAN-15	1267	100	10	1000	2	20	40	4	44	12 PETER	
9 01-JAN-15	1267	101	10	1000	2	30	60	6	66	12 PETER	
10 02-JAN-15	1268	100	11	2000	10	30	300	30	330	12 PETER	
11 09-FEB-15	1270	105	10	3000	20	70	1400	140	1540	12 PETER	
12 09-FEB-15	1270	106	10	3000	10	50	500	50	550	12 PETER	
13 10-FEB-15	1271	101	10	3000	10	30	300	30	330	12 PETER	
14 11-FEB-15	1272	200	12	4000	4	80	320	32	352	12 PETER	
15 11-FEB-15	1273	200	12	5000	6	80	480	48	528	12 PETER	
16 11-FEB-15	1274	100	10	6000	8	20	160	16	176	12 PETER	
17 11-FEB-15	1275	101	10	7000	10	30	300	30	330	12 PETER	
18 15-FEB-15	1276	100	11	8000	12	30	360	36	396	12 PETER	
19 16-FEB-15	1277	105	10	9000	14	70	980	98	1078	12 PETER	
20 16-FEB-15	1278	106	10	10000	16	50	800	80	880	12 PETER	
21 16-FEB-15	1279	101	10	11000	18	30	540	54	594	12 PETER	
22 16-FEB-15	1280	100	12	12000	20	80	1600	160	1760	12 PETER	

- u. Produce a list of rows with all columns from sales table for the sales generated by customers in North region

```
select * from sales s left join customer c on s.customer_id=s.customer_id where c.region = 'NORTH';
```

Database Management Systems Project

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Connections' (HR), 'SQL Worksheet' (selected), and 'History'. Below the tabs, there's a tree view of 'Oracle Connections' and 'Database Schema'. The main area is titled 'Worksheet' and contains a 'Query Builder' section with the following SQL code:

```
/* U */
select /* from sales s left|join customer c on s.customer_id=s.customer_id where c.region = 'NORTH';
```

Below the code, the results of the query are displayed in a table with 10 columns: SALES_DATE, ORDER_ID, PRODUCT_F_ID, CUSTOMER_F_ID, SALESPERSON_ID, QUANTITY, UNIT_PRICE, SALES_F_AMOUNT, TAX_AMOUNT, and TOTAL_AMOUNT. The table has 20 rows of data. The last row is highlighted in yellow.

SALES_DATE	ORDER_ID	PRODUCT_F_ID	CUSTOMER_F_ID	SALESPERSON_ID	QUANTITY	UNIT_PRICE	SALES_F_AMOUNT	TAX_AMOUNT	TOTAL_AMOUNT	CUSTOMER_ID_1	FIRST_J
1 12-JAN-15	1568	101	12	1000	20	400	400	40	440	12	PETER
2 15-JAN-15	1561	101	12	1000	20	400	400	40	440	12	PETER
3 16-JAN-15	1563	101	12	1000	20	400	400	40	440	12	PETER
4 12-JAN-15	1562	101	12	1000	20	400	400	40	440	12	PETER
5 12-JAN-15	1565	101	12	1000	20	400	400	40	440	12	PETER
6 01-FEB-15	1269	200	12	1000	20	80	1600	160	1760	12	PETER
7 01-FEB-15	1269	200	12	1000	20	80	1600	160	1760	12	PETER
8 01-JAN-15	1267	100	10	1000	2	20	40	4	44	12	PETER
9 01-JAN-15	1267	101	10	1000	2	30	60	6	66	12	PETER
10 02-JAN-15	1268	100	11	2000	10	30	3000	30	330	12	PETER
11 09-FEB-15	1278	105	10	3000	20	70	14000	140	1540	12	PETER
12 09-FEB-15	1278	106	10	3000	10	50	5000	50	550	12	PETER
13 10-FEB-15	1271	101	10	3000	10	30	3000	30	330	12	PETER
14 11-FEB-15	1272	200	12	4000	4	80	320	32	352	12	PETER
15 11-FEB-15	1273	200	12	5000	6	80	4000	48	528	12	PETER
16 11-FEB-15	1274	100	10	6000	8	20	1200	16	176	12	PETER
17 11-FEB-15	1275	101	10	7000	10	30	3000	30	330	12	PETER
18 15-FEB-15	1276	100	11	8000	12	30	3600	36	396	12	PETER
19 16-FEB-15	1277	105	10	9000	14	70	9800	98	1078	12	PETER
20 16-FEB-15	1278	106	10	10000	16	50	8000	80	880	12	PETER

- v. Produce a list of rows from the sales table where the selected total_amount is greater than the average total_amount of their respective customers

```
select * from sales_f e1
```

```
where e1.total_amount >
```

```
(select avg(e2.total_amount) from sales_f e2 where e1.customer_f_id=e2.customer_f_id
group by e2.customer_f_id);
```

SALES_F_DATE	ORDER_ID	PRODUCT_F_ID	CUSTOMER_F_ID	SALESPERSON_ID	QUANTITY	UNIT_PRICE	SALES_F_AMOUNT	TAX_AMOUNT	TOTAL_AMOUNT	CUSTOMER_ID_1	FIRST_J
25-FEB-15	1288	106	12	9000	36	80	2880	288	3168		
28-FEB-15	1289	101	12	10000	38	80	3040	304	3344		
02-MAR-15	1293	106	10	1000	46	70	3220	322	3542		
02-MAR-15	1294	101	10	2000	48	50	2400	240	2640		
07-MAR-15	1296	105	12	9000	52	80	4160	416	4576		
08-MAR-15	1297	106	12	10000	54	80	4320	432	4752		
08-MAR-15	1301	105	10	1000	34	70	2380	238	2618		
08-MAR-15	1302	106	10	1000	36	50	1800	180	1980		
15-MAR-15	1304	101	10	2000	40	80	3200	320	3520		
16-MAR-15	1305	100	12	3000	42	80	3360	336	3696		
19-MAR-15	1309	106	11	2000	50	70	3500	350	3850		
19-MAR-15	1310	101	12	3000	52	50	2600	260	2860		
23-MAR-15	1312	100	10	10000	34	80	2720	272	2992		

- w. Write a query to bring the names of the salespersons and their manager

```
select e1.first_name || "||" || e1.last_name as salespersons,e2.manager from
```

```
sales_fp e1 join sales_fp e2 on e1.manager=e2.first_name where e1.manager
is not null;
```

Database Management Systems Project

SALESPERSONS	MANAGER
SonuAgarwal	Tom
PatelHari	Tom
JamesGavin	Tom
RehmanAhmed	Raj
SaraK	Raj
RajKishore	-
TomJoseph	-
BobMoris	Jeff
GregChappel	Jeff
AnilKrishna	Raj
PeterMann	Jeff
JohnKing	Jeff

[Download CSV](#)

- x. Product the list of top 10 products based on the number of time they have been sold

```
select * from(
select distinct product_f_id, product_f_name, sum(sales_f.quantity) over(partition by
product_f_id)
total_sale from product_f join sales_f using (product_f_id) order by total_sale desc)
where rownum <=10;
```

PRODUCT_F_ID	PRODUCT_F_NAME	TOTAL_SALE
101	iPhone	1236
100	Mobile Cover	944
105	HTC 7800	440
106	Microsoft Keyboard 7865	412
200	Samsung F7100	50

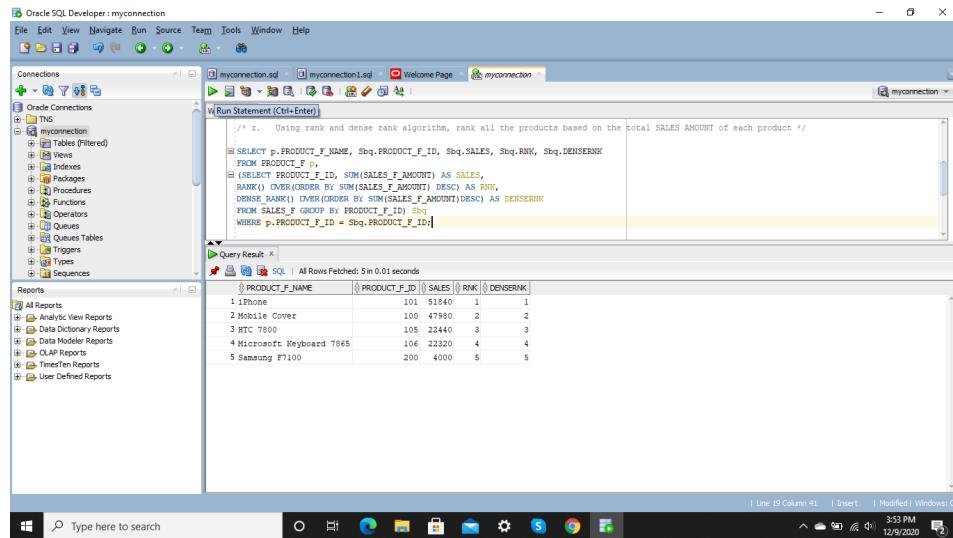
- y. List all the products who price is greater than the average price of all the products in the respective categories

```
select * from product_f e1
where e1.list_price >
(select avg(e2.list_price) from product_f e2 where
e1.product_f_category=e2.product_f_category group by e2.product_f_category);
```

PRODUCT_F_ID	PRODUCT_F_NAME	STANDARD_COST	COLOR	LIST_PRICE	PRODUCT_F_SIZE	WEIGHT	PRODUCT_F_CATEGORY
106	Microsoft Keyboard 7865	50	BLACK	60	3	20	Computer
101	iPhone	500	GOLD	600	6	20	Mobile

Database Management Systems Project

- z. Using rank and dense rank algorithm, rank all the products based on the total SALES AMOUNT of each product**



The screenshot shows the Oracle SQL Developer interface with a connection named "myconnection". In the central workspace, a query is being run:

```

/* z. Using rank and dense rank algorithm, rank all the products based on the total SALES AMOUNT of each product */
SELECT p.PRODUCT_F_NAME, Sbq.PRODUCT_F_ID, Sbq.SALES, Sbq.RNK,
Sbq.DENSERNK
FROM PRODUCT_F p,
(SELECT PRODUCT_F_ID,
SUM(SALES_F_AMOUNT) AS SALES,
RANK() OVER(ORDER BY SUM(SALES_F_AMOUNT) DESC) AS RNK,
DENSE_RANK() OVER(ORDER BY SUM(SALES_F_AMOUNT)DESC) AS DENSERNK
FROM SALES_F GROUP BY PRODUCT_F_ID) Sbq
WHERE p.PRODUCT_F_ID = Sbq.PRODUCT_F_ID;

```

The results are displayed in a table:

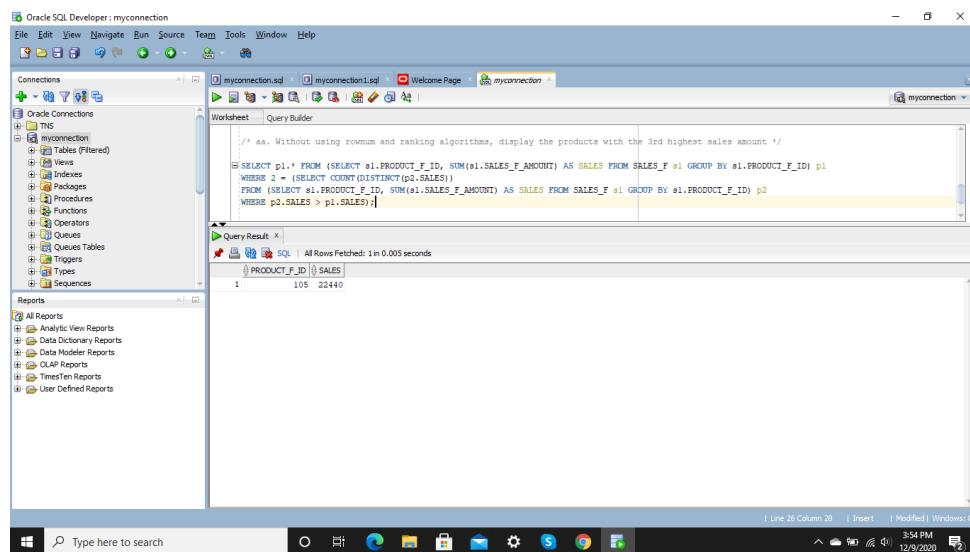
PRODUCT_F_NAME	PRODUCT_F_ID	SALES	RNK	DENSERNK
1 iPhone	101	51840	1	1
2 Mobile Cover	100	47980	2	2
3 HTC 7800	105	22440	3	3
4 Microsoft Keyboard 7865	106	22320	4	4
5 Samsung FT100	200	4000	5	5

```

SELECT p.PRODUCT_F_NAME, Sbq.PRODUCT_F_ID, Sbq.SALES, Sbq.RNK,
Sbq.DENSERNK
FROM PRODUCT_F p,
(SELECT PRODUCT_F_ID,
SUM(SALES_F_AMOUNT) AS SALES,
RANK() OVER(ORDER BY SUM(SALES_F_AMOUNT) DESC) AS RNK,
DENSE_RANK() OVER(ORDER BY SUM(SALES_F_AMOUNT)DESC) AS DENSERNK
FROM SALES_F GROUP BY PRODUCT_F_ID) Sbq
WHERE p.PRODUCT_F_ID = Sbq.PRODUCT_F_ID;

```

- aa. Without using rownum and ranking algorithms, display the products with the 3rd highest sales amount**



The screenshot shows the Oracle SQL Developer interface with a connection named "myconnection". In the central workspace, a query is being run:

```

/* aa. Without using rownum and ranking algorithms, display the products with the 3rd highest sales amount */
SELECT p1.* FROM (SELECT s1.PRODUCT_F_ID, SUM(s1.SALES_F_AMOUNT) AS SALES FROM SALES_F s1 GROUP BY s1.PRODUCT_F_ID) p1
WHERE 2 = (SELECT COUNT(DISTINCT p2.SALES)
FROM (SELECT s1.PRODUCT_F_ID, SUM(s1.SALES_F_AMOUNT) AS SALES FROM SALES_F s1 GROUP BY s1.PRODUCT_F_ID) p2
WHERE p2.SALES > p1.SALES);

```

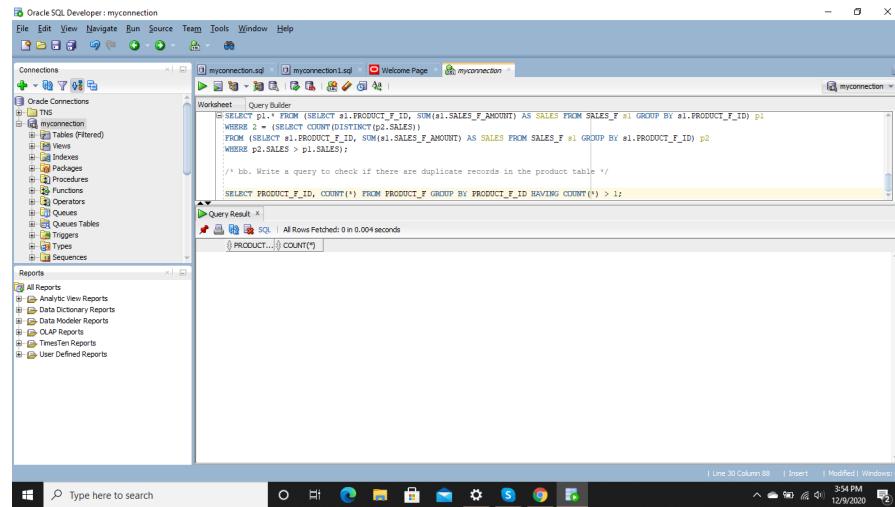
The results are displayed in a table:

PRODUCT_F_ID	SALES
105	22440

Database Management Systems Project

```
SELECT p1.* FROM (SELECT s1.PRODUCT_F_ID, SUM(s1.SALES_F_AMOUNT) AS SALES
FROM SALES_F s1 GROUP BY s1.PRODUCT_F_ID) p1 WHERE 2 = (SELECT
COUNT(DISTINCT(p2.SALES)) FROM (SELECT s1.PRODUCT_F_ID,
SUM(s1.SALES_F_AMOUNT) AS SALES FROM SALES_F s1 GROUP BY s1.PRODUCT_F_ID)
p2 WHERE p2.SALES > p1.SALES);
```

bb. Write a query to check if there are duplicate records in the product table



```
SELECT PRODUCT_F_ID, COUNT(*) FROM PRODUCT_F GROUP BY PRODUCT_F_ID
HAVING COUNT(*) > 1;
```