Lab 6
Subquery and Summary Query

1. Learning Objectives

After the successful completion of this lab, you will be able

- To explain what subqueries, summary queries, and Top-N queries are
- To read SQL statements that get information using subqueries and summary queries
- To write SQL statements that get information using summary queries and the following topics related to subquery
    - Top-N query, subquery in WHERE clauses and FROM clauses

2. Tasks to Complete

Complete the questions about subqueries and summary queries on the **MGS Database** included in the later part of this document. These queries use the tables in **user mgs**.

NOTE 1:

The main goal of this lab is for you to practice subquery. The 2nd goal is to practice summary queries.
So if your answer to each question does not use a subquery, you will get a 0 for that question.
Some of the questions require an answer using BOTH subquery and summary query.

NOTE 2:

The links to online Oracle SQL Language references are available in the in the Canvas Page: Links to Oracle SQL Language References.

RELATED KNOWLEDGE POINTS

- **A subquery is a SELECT statement inside another SELECT statement.**
- **A subquery often returns some data that is unknown in advance but is needed in the execution of the main query.**
- **A subquery is often used in these clauses of the main query:**
    - **FROM clause**
        - **Provide some data as temporary source table for the main query**
    - **WHERE clause**
        - **Provide some data used in limiting individual rows returned by the main query**
    - **HAVING clause**
        - **Provide some data used in limiting GROUPS of rows returned by the main query**
- **A Top-N query returns the top n rows or groups of rows based on the sorting criteria. It includes a subquery in a FROM clause of the main query. The subquery sorts the result in decreasing order. The main query also includes a WHERE clause with ROWNUM.**
- **When solving each question, it recommended that you follow this procedure**
    - Use pseudocode to outline the query.
    - If necessary, use pseudocode to outline each subquery.
    - Code the subqueries and test them.
    - Code and test the final query.

3. Submission Requirements

WHEN

- **Please see the lab Canvas page for details**.

WHAT

- A text file with the extension **.sql, including all your SQL statements.**
- The file is in the following format**.**

Mark each query based on the question number. Write your FULL name on the first page.
Sample:

```
--Lab6
--Your full name

--Q1
Your solution.
--Q2
Your solution.
…
```

HOW

- Submit your SQL script file by attaching it to the link **Lab 6** in folder **Assignments\Labs** on Canvas.

4. Grading

This assignment is graded based on the CORRECTNESS of YOUR ANSWER.

There are 8 questions. Each question counts 12.5 points.

If your answer to a question is not 100% correct, you will get partial credits.

Queries Containing Subquery on MGS Database

1. Write a SELECT statement that is equivalent to the SELECT statement below, but don't use a join. Instead, use a SUBQUERY in a WHERE clause that uses the IN operator.

```
SELECT DISTINCT category_name
FROM categories c JOIN products p
  ON c.category_id = p.category_id
ORDER BY category_name;
```

HINT:

- **Think about what information is unknown but used in the final result.**
- **See Q1 in In-Class Practice.**

OUTPUT

	CATEGORY_NAME
1	Basses
2	Drums
3	Guitars

2. Print the product name and actual price of products whose actual price is above the average actual price of all products. The actual price is the price after the discount.

Use the heading ACTUAL_PRICE for the actual prices in the result.
Sort the result such that the product with the highest actual price appears first.
You MUST use SUBQUERY.

HINT:

- **Use a subquery to get the average actual price of all products.**
- **See Q2 in In-Class Practice.**

OUTPUT

	PRODUCT_NAME	ACTUAL_PRICE
1	Gibson SG	1208.16
2	Gibson Les Paul	839.3
3	Tama 5-Piece Drum Set with Cymbals	679.9915
4	Fender Precision	559.993

3. Print the ids and names of all categories that currently don't have any product.
You MUST use a SUBQUERY and the NOT IN operator.

HINT:

- **Use a subquery to get the list of all categories with a product.**
- **See Q3 in In-Class Practice.**

OUTPUT

	CATEGORY_ID	CATEGORY_NAME
1	4	Keyboards

4. Print the ids, codes, product names, and actual prices of the top 3 products that are most expensive based on the actual price, which is the price after the discount.
Use the heading ACTUAL_PRICE for the actual prices in the result.
You MUST use a SUBQUERY in the FROM clause in your answer.
HINT:

- **This is a Top-N query.**
- **See the KNOWLEDGE POINTS IN PAGE 1.**
- **See Q5 in IN-CLASS Practice.**

OUTPUT

	PRODUCT_ID	PRODUCT_CODE	PRODUCT_NAME	ACTUAL_PRICE
1	3	sg	Gibson SG	1208.16
2	2	les_paul	Gibson Les Paul	839.3
3	10	tama	Tama 5-Piece Drum Set with Cymbals	679.9915

5. Print the ids, codes, product names, **category names**, and actual prices of the top 3 products that are most expensive based on the actual price, which is the price after the discount.
Use the heading ACTUAL_PRICE for the actual prices in the result.
You MUST use a SUBQUERY in the FROM clause in your answer.
HINT:

- **This is a Top-N query.**
- **See the KNOWLEDGE POINTS IN PAGE 1.**

- See Q5 in IN-CLASS Practice.
- You need a JOIN in your subquery.

OUTPUT

	PRODUCT_ID	PRODUCT_CODE	PRODUCT_NAME	CATEGORY_NAME	ACTUAL_PRICE
1	3	sg	Gibson SG	Guitars	1208.16
2	2	les_paul	Gibson Les Paul	Guitars	839.3
3	10	tama	Tama 5-Piece Drum Set with Cymbals	Drums	679.9915

6. Print the category ids and product counts of the top 3 categories with the most products.
You MUST use a SUBQUERY in the FROM clause in your answer.
You need an aggregation function too.
HINT:
   - This is a Top-N query.
   - See the KNOWLEDGE POINTS IN PAGE 1.
   - See Q5 in IN-CLASS Practice.
   - You need an aggregation function in your subquery to get the number of products in each category.

OUTPUT

	CATEGORY_ID	PRODUCT_COUNT
1	1	6
2	2	2
3	3	2

7. Print the name and discount percent of each product that has a unique discount percent.
In other words, don't include products that have the same discount percent as another product.
Sort the result in the increasing alphabetical order of product names.
You MUST use a SUBQUERY in your answer.
You need an aggregation function too.
HINT:
   - In the subquery, count the number of products in each discount percentage and return only those discount percentages with product count being 1.
   - You need an aggregation function in your subquery.
   - Your subquery need to be in the WHERE clause of the main query.

OUTPUT

	PRODUCT_NAME	DISCOUNT_PERCENT
1	Gibson SG	52
2	Hofner Icon	25
3	Rodriguez Caballero 11	39
4	Tama 5-Piece Drum Set with Cymbals	15
5	Washburn D10S	0
6	Yamaha FG700S	38

8.

a.

Write a query that prints the email_address of each customer who placed an order, and the order_id and order total of each order placed by this customer.
Sort the result in the increasing alphabetical order of email addresses.
The order total is the total actual prices of all items in an order. The actual price of an item is the price after the discount. The email addresses of customers are unique.

HINT

- **To do this, you can group the result set by the email_address and order_id columns. In addition, you must calculate the order total from the columns in the Order_Items table.**
- NOTE: This FIRST query does not have to contain a subquery. But it needs an aggregation function.

OUTPUT

	EMAIL_ADDRESS	ORDER_ID	ORDER_TOTAL
1	allan.sherwood@yahoo.com	1	839.3
2	allan.sherwood@yahoo.com	3	1461.31
3	barryz@gmail.com	2	303.79
4	christineb@solarone.com	4	1678.6
5	david.goldstein@hotmail.com	5	299
6	david.goldstein@hotmail.com	9	489.3
7	erinv@gmail.com	6	299
8	frankwilson@sbcglobal.net	7	1539.28
9	gary_hernandez@yahoo.com	8	679.99

b.
Write a SECOND query that USEs the FIRST query in its FROM clause.
The main query returns the email address of each customer and the largest order total among all orders placed by this customer.
Use the heading MAX_ORDER_TOTAL for the largest order totals in the result.
Sort the result in the increasing alphabetical order of email addresses.
HINT

- **To do this, you can group the result set by the email_address column.**

OUTPUT

	EMAIL_ADDRESS	MAX_ORDER_TOTAL
1	allan.sherwood@yahoo.com	1461.31
2	barryz@gmail.com	303.79
3	christineb@solarone.com	1678.6
4	david.goldstein@hotmail.com	489.3
5	erinv@gmail.com	299
6	frankwilson@sbcglobal.net	1539.28
7	gary_hernandez@yahoo.com	679.99

Remember that you need to SUBMIT TWO queries for Question 8.