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

Practical Assignment #4
“Database Management Systems”
SQL Constructs

Author: Emir Oğuz
Course, Group: DSP201, Group 1
Student Card No: 230ADB011

Checked: Andrejs Gaidukovs

2022 / 2023 Study Year

Content

1 Goal

Learn about and use various SQL constructs.

Queries codes should be inserted in MS word report like text, not screenshot in picture format. Take your script from previous works and add queries to this script (continues work from previous HAs).

Submit to ORTUS:
1) MS Word report file names: DBMS_4_YourSurname.docx ;
2) Combined script (including all previous HAs) in .sql format : DBMS_4_YourSurname.sql ;

2 Task

1. Create all tables (and all related objects) and fill tables with data, at least 20 records in every table;

2. Create the following types of SQL queries (at least 10 queries):

   1) query with one table and multiple conditions in **WHERE** clause;
   2) query with two related tables and multiple conditions in **WHERE** clause;
   3) query with **GROUP BY** grouping;
   4) query with two related tables and query with **GROUP BY** grouping;
   5) query with grouping and **HAVING** clause (conditions for groups);
   6) query with two related tables, grouping and **HAVING** clause (conditions for groups);
   7) query with subquery in **SELECT** clause;
   8) query with subquery in **FROM** clause;
   9) query with subquery in **WHERE** clause;
   10) query with subquery in **HAVING** clause;
   11) query with **UNION, INTERSECT, MINUS** constructions;
   12) query with **EXISTS** construction;
   13) at least 3 queries with **INNER JOIN**;
   14) at least 3 queries with **LEFT JOIN**;
   15) at least 3 queries with **RIGHT JOIN**;

   In queries include:

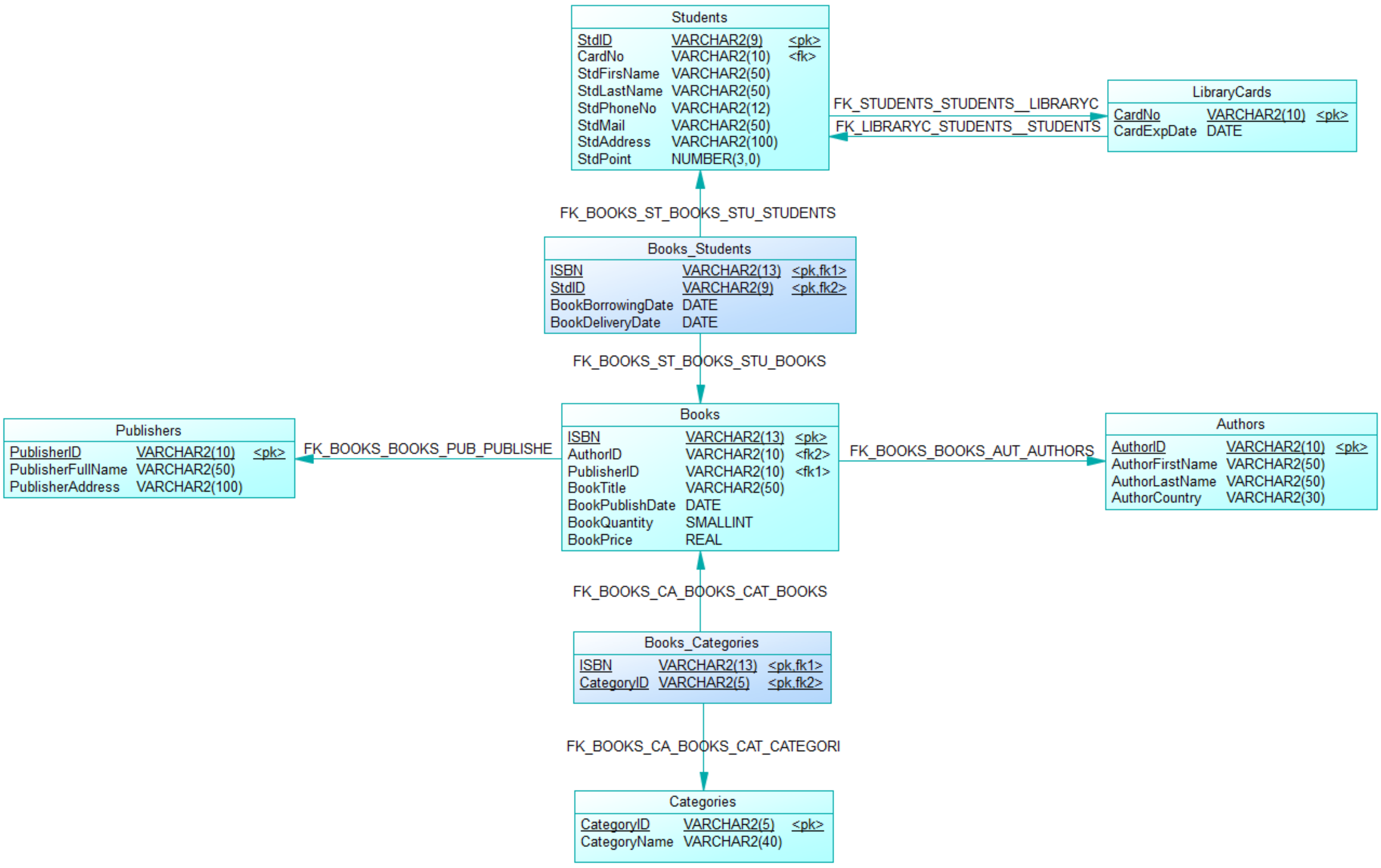
   - **LIKE** together with **"?", "*", &;**
   - **ANY**; **ALL; IN; BETWEEN; DISTINCT**.
   - **TOP (FETCH)**

   In queries include aggregation functions **COUNT(), SUM(), MAX(), AVG()**.

   In queries include text and datetime functions.

3. Conclusions

3 Database Description

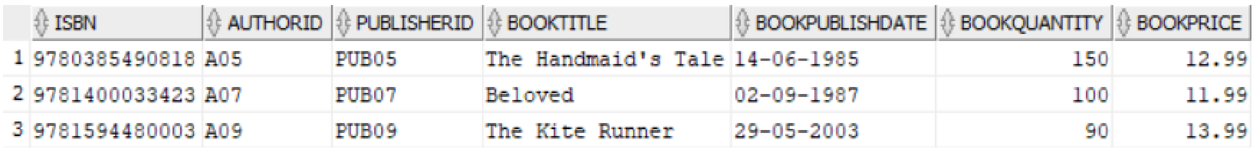

Students
StdID VARCHAR2(9) <pk>
CardNo VARCHAR2(10) <fk>
StdFirsName VARCHAR2(50)
StdLastName VARCHAR2(50)
StdPhoneNo VARCHAR2(12)
StdMail VARCHAR2(50)
StdAddress VARCHAR2(100)
StdPoint NUMBER(3,0)
FK_STUDENTS_STUDENTS__LIBRARYC
FK_LIBRARYC_STUDENTS__STUDENTS
LibraryCards
CardNo VARCHAR2(10) <pk>
CardExpDate DATE
FK_BOOKS_ST_BOOKS_STU_STUDENTS
Books_Students
ISBN VARCHAR2(13) <pk,fk1>
StdID VARCHAR2(9) <pk,fk2>
BookBorrowingDate DATE
BookDeliveryDate DATE
FK_BOOKS_ST_BOOKS_STU_BOOKS
Publishers
PublisherID VARCHAR2(10) <pk>
PublisherFullName VARCHAR2(50)
PublisherAddress VARCHAR2(100)
FK_BOOKS_BOOKS_PUB_PUBLISHE
Books
ISBN VARCHAR2(13) <pk>
AuthorID VARCHAR2(10) <fk2>
PublisherID VARCHAR2(10) <fk1>
BookTitle VARCHAR2(50)
BookPublishDate DATE
BookQuantity SMALLINT
BookPrice REAL
FK_BOOKS_BOOKS_AUT_AUTHORS
Authors
AuthorID VARCHAR2(10) <pk>
AuthorFirstName VARCHAR2(50)
AuthorLastName VARCHAR2(50)
AuthorCountry VARCHAR2(30)
FK_BOOKS_CA_BOOKS_CAT_BOOKS
Books_Categories
ISBN VARCHAR2(13) <pk,fk1>
CategoryID VARCHAR2(5) <pk,fk2>
FK_BOOKS_CA_BOOKS_CAT_CATEGORI
Categories
CategoryID VARCHAR2(5) <pk>
CategoryName VARCHAR2(40)

4 SQL Queries

4.1 Query #1 One Table and Multiple Conditions in WHERE Clause

- This query retrieves all columns from the BOOKS table where the book quantity is greater than 80, the book price is less than 17.00 and books with a publishing date after January 1, 1970.

```
SELECT * FROM BOOKS
WHERE BOOKPUBLISHDATE > TO_DATE('1970/01/01', 'YYYY/MM/DD')
  AND BOOKQUANTITY > 80
  AND BOOKPRICE < 17.00;
```

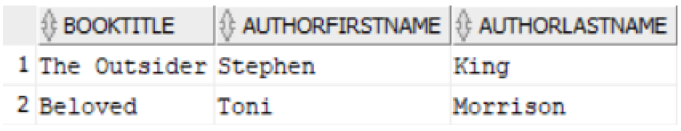


	ISBN	AUTHORID	PUBLISHERID	BOOKTITLE	BOOKPUBLISHDATE	BOOKQUANTITY	BOOKPRICE
1	9780385490818	A05	PUB05	The Handmaid's Tale	14-06-1985	150	12.99
2	9781400033423	A07	PUB07	Beloved	02-09-1987	100	11.99
3	9781594480003	A09	PUB09	The Kite Runner	29-05-2003	90	13.99

4.2 Query #2 Two Related Tables and Multiple Conditions in WHERE Clause

- This query selects the book name (BOOKTITLE) from the BOOKS table, the author name (AUTHORFIRSTNAME) and the author surname (AUTHORLASTNAME) from the AUTHORS table. It uses the INNER JOIN expression to join the two tables based on the common column AUTHORID. The WHERE clause includes multiple conditions: it filters for books published on or after January 1, 1980 (BOOKPUBLISHDATE >= TO_DATE('1980/01/01', 'YYYY/MM/DD')) and authors from the United States (A.AUTHORCOUNTRY = 'United States'). Feel free to modify the conditions or select different columns based on your specific requirements.

```
SELECT B.BOOKTITLE, A.AUTHORFIRSTNAME, A.AUTHORLASTNAME
FROM BOOKS B
INNER JOIN AUTHORS A ON B.AUTHORID = A.AUTHORID
WHERE B.BOOKPUBLISHDATE >= TO_DATE('1980/01/01', 'YYYY/MM/DD')
   AND A.AUTHORCOUNTRY = 'United States';
```

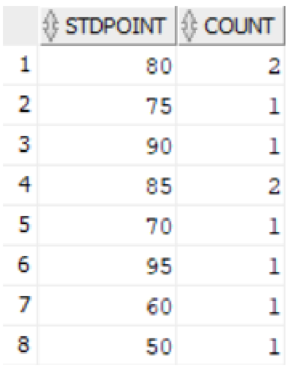


	BOOKTITLE	AUTHORFIRSTNAME	AUTHORLASTNAME
1	The Outsider	Stephen	King
2	Beloved	Toni	Morrison

4.3 Query #3 Grouping with GROUP BY

- This query selects the "STDPOINT" column from the "STUDENTS" table and counts the number of students in each grade using the COUNT(*) function. The "GROUP BY" clause is used to group the results by the "STDPOINT" column. This query will provide you with the count of students for each unique grade in the table.

```
SELECT STDPOINT, COUNT(*) AS COUNT
FROM STUDENTS
GROUP BY STDPOINT;
```

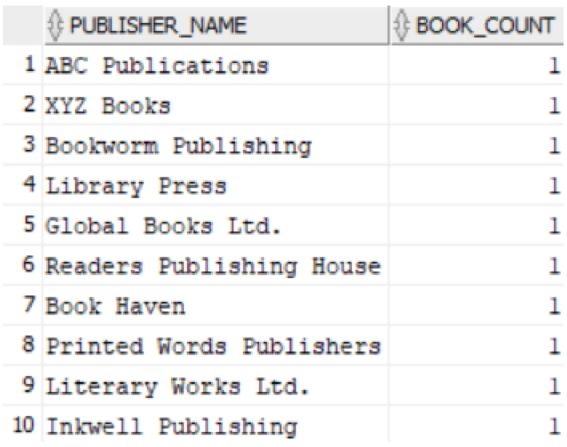


	STDPOINT	COUNT
1	80	2
2	75	1
3	90	1
4	85	2
5	70	1
6	95	1
7	60	1
8	50	1

4.4 Query #4 Two Related Tables and GROUP BY Grouping

- In this query, we are selecting the "PUBLISHERFULLNAME" column from the "PUBLISHERS" table as "PUBLISHER_NAME" and counting the number of books associated with each publisher using the COUNT(B.ISBN) function. We join the "PUBLISHERS" and "BOOKS" tables using the common column "PUBLISHERID" from "PUBLISHERS" and "PUBLISHERID" from "BOOKS." Finally, we group the results by the publisher's name using the "GROUP BY" clause. This query will provide you with the publisher name and the count of books associated with each publisher.

```
SELECT P.PUBLISHERFULLNAME AS PUBLISHER_NAME, COUNT(B.ISBN) AS
BOOK_COUNT
FROM PUBLISHERS P
INNER JOIN BOOKS B ON P.PUBLISHERID = B.PUBLISHERID
GROUP BY P.PUBLISHERFULLNAME;
```

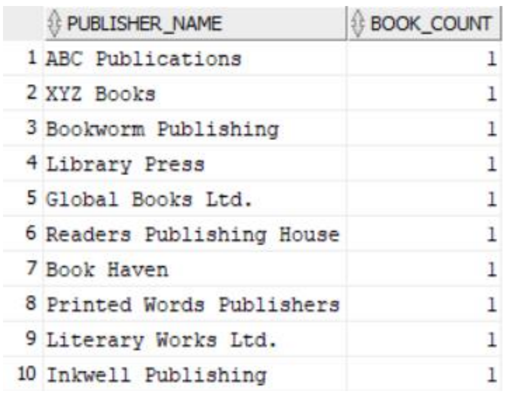


	PUBLISHER_NAME	BOOK_COUNT
1	ABC Publications	1
2	XYZ Books	1
3	Bookworm Publishing	1
4	Library Press	1
5	Global Books Ltd.	1
6	Readers Publishing House	1
7	Book Haven	1
8	Printed Words Publishers	1
9	Literary Works Ltd.	1
10	Inkwell Publishing	1

4.5 Query #5 Grouping and HAVING Clause (Conditions for Groups)

- In this query, we are selecting the "PUBLISHERFULLNAME" column from the "PUBLISHERS" table as "PUBLISHER_NAME" and counting the number of books associated with each publisher using the COUNT(B.ISBN) function. We join the "PUBLISHERS" and "BOOKS" tables using the common column "PUBLISHERID" from "PUBLISHERS" and "BOOKS." The results are then grouped by the publisher's name using the "GROUP BY" clause.

```
SELECT P.PUBLISHERFULLNAME AS PUBLISHER_NAME, COUNT(B.ISBN) AS
BOOK_COUNT
FROM PUBLISHERS P
INNER JOIN BOOKS B ON P.PUBLISHERID = B.PUBLISHERID
GROUP BY P.PUBLISHERFULLNAME
HAVING COUNT(B.ISBN) = 1;
```

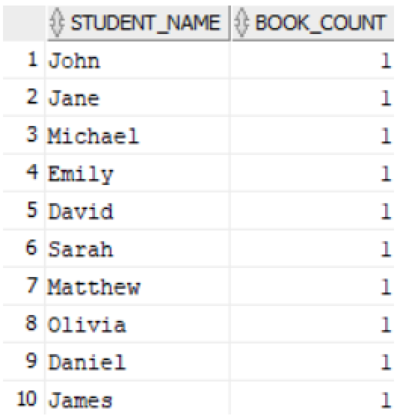


	PUBLISHER_NAME	BOOK_COUNT
1	ABC Publications	1
2	XYZ Books	1
3	Bookworm Publishing	1
4	Library Press	1
5	Global Books Ltd.	1
6	Readers Publishing House	1
7	Book Haven	1
8	Printed Words Publishers	1
9	Literary Works Ltd.	1
10	Inkwell Publishing	1

4.6 Query #6 Two Related Tables, Grouping and HAVING Clause (Conditions for Groups)

- In this query, we are selecting the "STDFIRSTNAME" column from the "STUDENTS" table as "STUDENT_NAME" and counting the number of books associated with each student using the COUNT(BS.BOOK_ID) function. We join the "STUDENTS" and "BOOKS_STUDENTS" tables using the common column "STDID" from "STUDENTS" and "BOOKS_STUDENTS." The results are then grouped by the student's name using the "GROUP BY" clause.

```
SELECT S.STDFIRSNAME AS STUDENT_NAME, COUNT(BS.ISBN) AS BOOK_COUNT
FROM STUDENTS S
INNER JOIN BOOKS_STUDENTS BS ON S.STDID = BS.STDID
GROUP BY S.STDFIRSNAME
HAVING COUNT(BS.ISBN) >= 1;
```

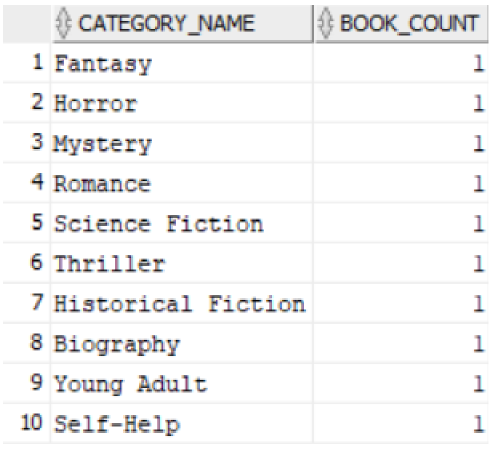


	STUDENT_NAME	BOOK_COUNT
1	John	1
2	Jane	1
3	Michael	1
4	Emily	1
5	David	1
6	Sarah	1
7	Matthew	1
8	Olivia	1
9	Daniel	1
10	James	1

4.7 Query #7 Subquery in SELECT Clause

- In this query, we are selecting the "CATEGORYNAME" column from the "CATEGORIES" table as "CATEGORY_NAME". Inside the SELECT clause, we have a subquery that counts the number of books associated with each category. The subquery is executed for each row in the outer query.

```
SELECT
   C.CATEGORYNAME AS CATEGORY_NAME,
   (SELECT COUNT(*) FROM BOOKS_CATEGORIES BC WHERE BC.CATEGORYID =
C.CATEGORYID) AS BOOK_COUNT
FROM
   CATEGORIES C;
```

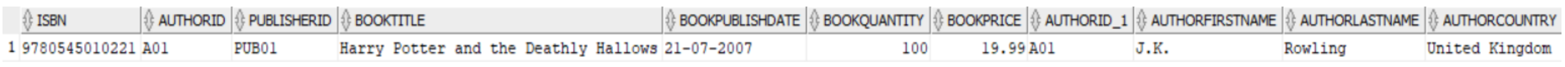


	CATEGORY_NAME	BOOK_COUNT
1	Fantasy	1
2	Horror	1
3	Mystery	1
4	Romance	1
5	Science Fiction	1
6	Thriller	1
7	Historical Fiction	1
8	Biography	1
9	Young Adult	1
10	Self-Help	1

4.8 Query #8 Subquery in FROM Clause

- This query will use a subquery in the FROM clause to select all books written by authors from the United Kingdom. The subquery selects the AUTHORID column from the AUTHORS table where the AUTHORCOUNTRY is equal to 'United Kingdom', and the WHERE clause in the outer query filters the BOOKS table to only include books written by those authors. The result of the subquery is then used as a derived table labeled as UK_BOOKS.

```
SELECT *
FROM (
   SELECT *
   FROM BOOKS
   WHERE AUTHORID IN (
      SELECT AUTHORID
      FROM AUTHORS
      WHERE AUTHORCOUNTRY = 'United Kingdom'
   )
) UK_BOOKS
INNER JOIN AUTHORS ON UK_BOOKS.AUTHORID = AUTHORS.AUTHORID;
```

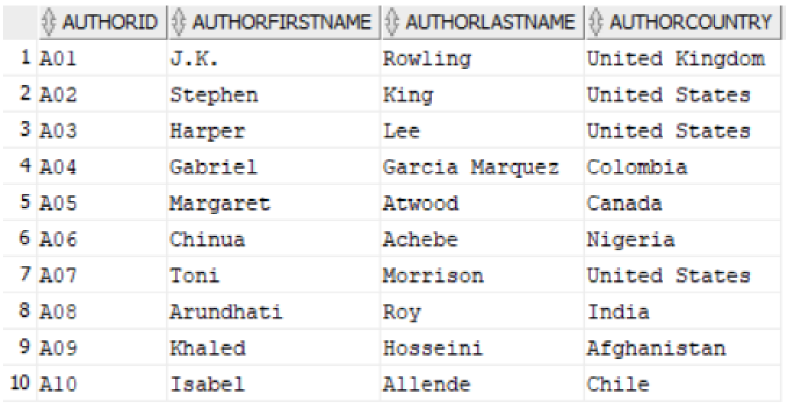


	ISBN	AUTHORID	PUBLISHERID	BOOKTITLE	BOOKPUBLISHDATE	BOOKQUANTITY	BOOKPRICE	AUTHORID_1	AUTHORFIRSTNAME	AUTHORLASTNAME	AUTHORCOUNTRY
1	9780545010221	A01	PUB01	Harry Potter and the Deathly Hallows	21-07-2007	100	19.99	A01	J.K.	Rowling	United Kingdom

4.9 Query #9 Subquery in WHERE Clause

- This query will select all authors from the AUTHORS table whose AUTHORID is included in a subquery that selects the AUTHORID column from the BOOKS table, and groups the results by AUTHORID.

```
SELECT *
FROM AUTHORS
WHERE AUTHORID IN (
   SELECT AUTHORID
   FROM BOOKS
   GROUP BY AUTHORID
);
```

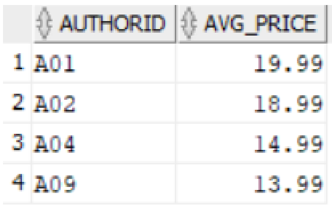


	AUTHORID	AUTHORFIRSTNAME	AUTHORLASTNAME	AUTHORCOUNTRY
1	A01	J.K.	Rowling	United Kingdom
2	A02	Stephen	King	United States
3	A03	Harper	Lee	United States
4	A04	Gabriel	Garcia Marquez	Colombia
5	A05	Margaret	Atwood	Canada
6	A06	Chinua	Achebe	Nigeria
7	A07	Toni	Morrison	United States
8	A08	Arundhati	Roy	India
9	A09	Khaled	Hosseini	Afghanistan
10	A10	Isabel	Allende	Chile

4.10 Query #10 Subquery in HAVING Clause

- This query will join the BOOKS and AUTHORS tables on the AUTHORID column, group the results by AUTHORID, and calculate the average price of books for each author. The HAVING clause includes a subquery that calculates the overall average book price for the entire BOOKS table, and only includes those authors whose average book price is greater than the overall average price.

```
SELECT AUTHORS.AUTHORID, AVG(BOOKS.BOOKPRICE) AS AVG_PRICE
FROM BOOKS
INNER JOIN AUTHORS ON BOOKS.AUTHORID = AUTHORS.AUTHORID
GROUP BY AUTHORS.AUTHORID
HAVING AVG(BOOKS.BOOKPRICE) > (
   SELECT AVG(BOOKPRICE)
   FROM BOOKS
);
```



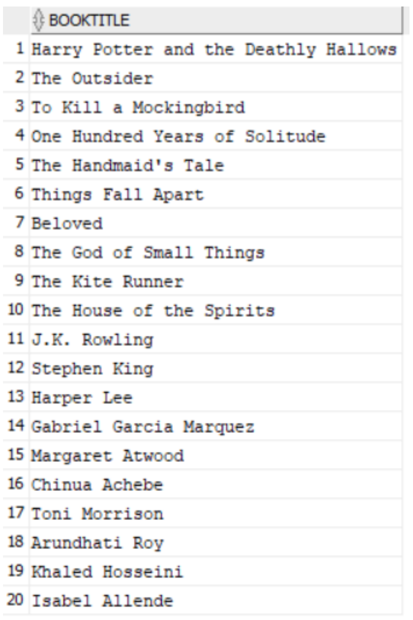
	AUTHORID	AVG_PRICE
1	A01	19.99
2	A02	18.99
3	A04	14.99
4	A09	13.99

4.11 Query #11 UNION, INTERSECT, MINUS Constructions

1) Union

- This query uses the UNION operator to combine the results of two separate queries into a single result set.

```
SELECT BOOKTITLE FROM BOOKS
UNION
SELECT AUTHORFIRSTNAME || ' ' || AUTHORLASTNAME FROM AUTHORS;
```

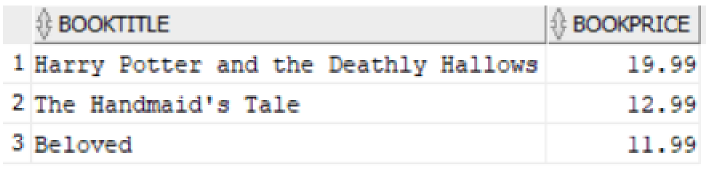


	BOOKTITLE
1	Harry Potter and the Deathly Hallows
2	The Outsider
3	To Kill a Mockingbird
4	One Hundred Years of Solitude
5	The Handmaid's Tale
6	Things Fall Apart
7	Beloved
8	The God of Small Things
9	The Kite Runner
10	The House of the Spirits
11	J.K. Rowling
12	Stephen King
13	Harper Lee
14	Gabriel Garcia Marquez
15	Margaret Atwood
16	Chinua Achebe
17	Toni Morrison
18	Arundhati Roy
19	Khaled Hosseini
20	Isabel Allende

2) Intersect

- This query will select all books from the BOOKS table where the BOOKPRICE is greater than 11.00, and return only those books that also have a BOOKQUANTITY greater than 95, using the INTERSECT operator.

```
SELECT BOOKTITLE, BOOKPRICE
FROM BOOKS
WHERE BOOKPRICE > 11.00
INTERSECT
SELECT BOOKTITLE, BOOKPRICE
FROM BOOKS
WHERE BOOKQUANTITY > 95;
```

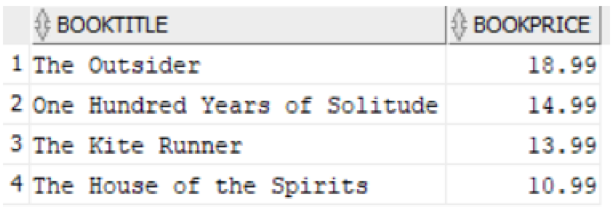


	BOOKTITLE	BOOKPRICE
1	Harry Potter and the Deathly Hallows	19.99
2	The Handmaid's Tale	12.99
3	Beloved	11.99

3) Minus

- This query will select all books from the BOOKS table where the BOOKPRICE is greater than 10.00, and then remove any books from that set that also have a BOOKQUANTITY greater than or equal to 100, using the MINUS operator.

```
SELECT BOOKTITLE, BOOKPRICE
FROM BOOKS
WHERE BOOKPRICE > 10.00
MINUS
SELECT BOOKTITLE, BOOKPRICE
FROM BOOKS
WHERE BOOKQUANTITY >= 100;
```

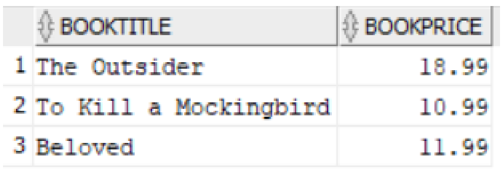


	BOOKTITLE	BOOKPRICE
1	The Outsider	18.99
2	One Hundred Years of Solitude	14.99
3	The Kite Runner	13.99
4	The House of the Spirits	10.99

4.12 Query #12 EXISTS Construction

- This query will select all books from the BOOKS table that have an associated author whose country is 'United States'.

```
SELECT BOOKTITLE, BOOKPRICE
FROM BOOKS b
WHERE EXISTS (
  SELECT 1
  FROM AUTHORS a
  WHERE a.AUTHORID = b.AUTHORID
    AND a.AUTHORCOUNTRY = 'United States'
);
```

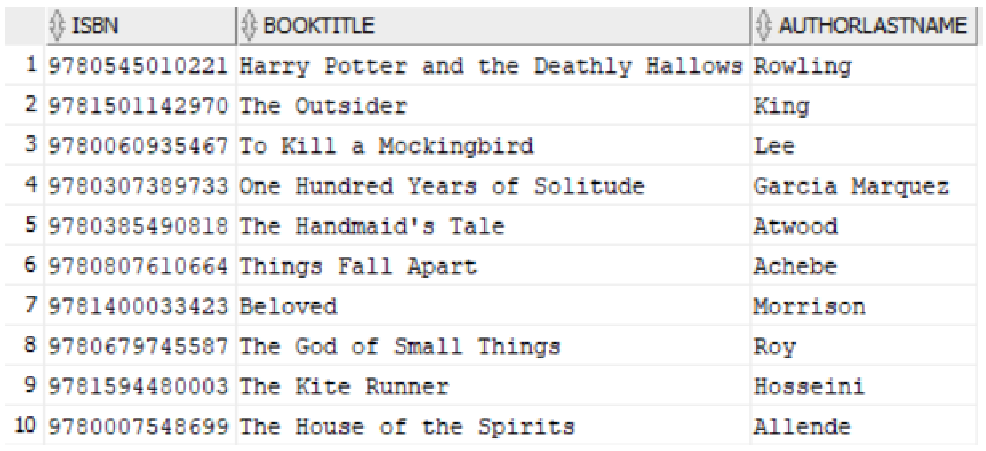


	BOOKTITLE	BOOKPRICE
1	The Outsider	18.99
2	To Kill a Mockingbird	10.99
3	Beloved	11.99

4.13 Query #13 LEFT JOIN

- This query selects the ISBN, book title, and author last name for all books in the BOOKS table, including those that do not have a corresponding author in the AUTHORS table.

```
SELECT BOOKS.ISBN, BOOKS.BOOKTITLE, AUTHORS.AUTHORLASTNAME
FROM BOOKS
LEFT JOIN AUTHORS ON BOOKS.AUTHORID = AUTHORS.AUTHORID;
```

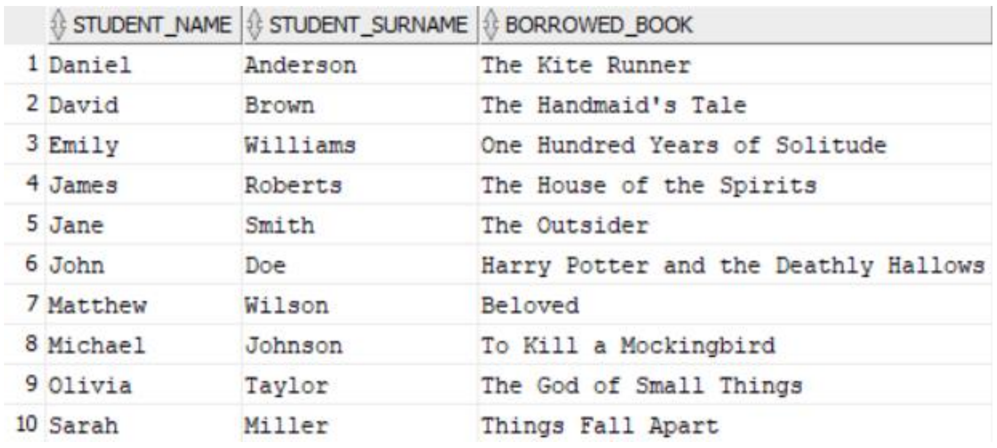


	ISBN	BOOKTITLE	AUTHORLASTNAME
1	9780545010221	Harry Potter and the Deathly Hallows	Rowling
2	9781501142970	The Outsider	King
3	9780060935467	To Kill a Mockingbird	Lee
4	9780307389733	One Hundred Years of Solitude	Garcia Marquez
5	9780385490818	The Handmaid's Tale	Atwood
6	9780807610664	Things Fall Apart	Achebe
7	9781400033423	Beloved	Morrison
8	9780679745587	The God of Small Things	Roy
9	9781594480003	The Kite Runner	Hosseini
10	9780007548699	The House of the Spirits	Allende

4.14 Query #14 RIGHT JOIN and LEFT JOIN

- In this query, we are selecting the "STDFIRSTNAME" and “STDLASTNAME” columns from the "STUDENTS" table as "STUDENT_NAME" and “STUDENT_SURNAME” and using the COALESCE function to handle cases where a student doesn't have any books assigned.

```
SELECT
   S.STDFIRSTNAME AS STUDENT_NAME,
   S.STDLASTNAME AS STUDENT_SURNAME,
   COALESCE(B.BOOKTITLE, 'No Book Assigned') AS BORROWED_BOOK
FROM STUDENTS S
RIGHT JOIN BOOKS_STUDENTS BS ON S.STDID = BS.STDID
LEFT JOIN BOOKS B ON BS.ISBN = B.ISBN
ORDER BY STUDENT_NAME;
```

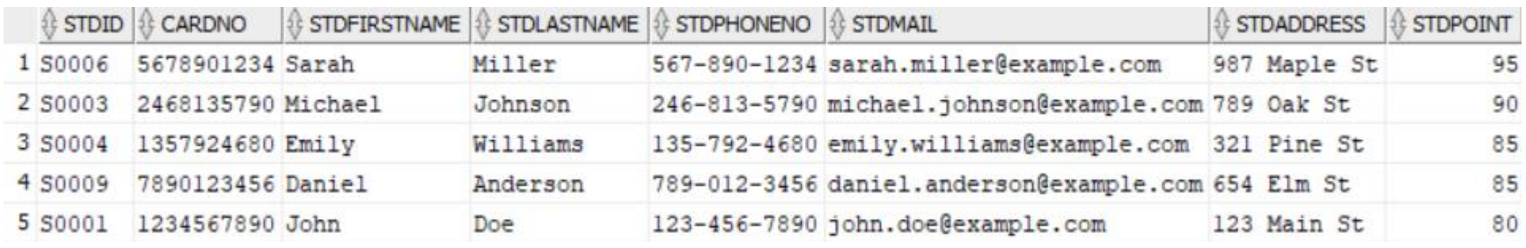


	STUDENT_NAME	STUDENT_SURNAME	BORROWED_BOOK
1	Daniel	Anderson	The Kite Runner
2	David	Brown	The Handmaid's Tale
3	Emily	Williams	One Hundred Years of Solitude
4	James	Roberts	The House of the Spirits
5	Jane	Smith	The Outsider
6	John	Doe	Harry Potter and the Deathly Hallows
7	Matthew	Wilson	Beloved
8	Michael	Johnson	To Kill a Mockingbird
9	Olivia	Taylor	The God of Small Things
10	Sarah	Miller	Things Fall Apart

4.15 Query #15 FETCH and ORDER BY

- This query retrieves information from the "STUDENTS" table. It selects the student ID, first name, last name, phone number, email, address, and points for each student. The results are ordered in descending order based on the student's points. Finally, it fetches only the first 10 rows, providing the top 10 students with the highest points.

```
SELECT * FROM STUDENTS
ORDER BY STDPOINT DESC
FETCH FIRST 5 ROWS ONLY;
```



	STDID	CARDNO	STDFIRSTNAME	STDLASTNAME	STDPHONENO	STDMAIL	STDADDRESS	STDPOINT
1	S0006	5678901234	Sarah	Miller	567-890-1234	sarah.miller@example.com	987 Maple St	95
2	S0003	2468135790	Michael	Johnson	246-813-5790	michael.johnson@example.com	789 Oak St	90
3	S0004	1357924680	Emily	Williams	135-792-4680	emily.williams@example.com	321 Pine St	85
4	S0009	7890123456	Daniel	Anderson	789-012-3456	daniel.anderson@example.com	654 Elm St	85
5	S0001	1234567890	John	Doe	123-456-7890	john.doe@example.com	123 Main St	80

5 Conclusions

Criteria	Minimum	Count of Queries	Number of Queries
Query with 2 or more conditions	2	3	4.1, 4.2, 4.12
From with 2 or more tables	10	12	4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12
Use of *Group by*	1	6	4.3, 4.4, 4.5, 4.6, 4.9, 4.10
From with 2 or more tables and *Group by*	1	1	4.9
Use of *Having*	1	3	4.5, 4.6, 4.10
From with 2 or more tables and *Having*	1	1	4.10
Ordering of data	2	2	4.14, 4.15
query with subquery in SELECT clause;	1	1	4.7
query with subquery in FROM clause;	1	1	4.8
query with subquery in WHERE clause;	1	1	4.9
query with UNION construction;	1	1	4.11
query with EXISTS construction;	1	1	4.12
queries with INNER JOIN;	3	6	4.2, 4.4, 4.5, 4.6, 4.8, 4.10
queries with LEFT JOIN;	3	2	4.13, 4.14
queries with RIGHT JOIN;	3	1	4.14
Use of aggregation functions	3	6	4.3, 4.4, 4.5, 4.6, 4.7, 4.10
Use of text functions	1	1	4.14
Use of datetime functions	1	2	4.1, 4.2
Use of FETCH predicate	1	1	4.15
Use of expressions (INTERSECT, MINUS)	1	1	4.11

In this task, I worked with a sample SQL script and database and used various SQL commands to query and manipulate data. I started by creating tables, adding constraints and indexes to enforce data integrity, and populating the tables with sample data. Then I used basic SQL queries to retrieve data from the tables, including SELECT, WHERE, and ORDER BY statements.

Next, I explored more advanced SQL concepts, such as JOINs, GROUP BY, HAVING, subqueries, and set operations like UNION and INTERSECT. I also used date-time functions like TO_DATE to modify and format our query results.

Through these exercises, I learned how to extract specific information from a database by querying multiple tables, grouping data, and applying various conditions. These skills are essential for data analysts and developers who need to interact with databases on a regular basis. With practice, I can become more proficient in SQL and use it to extract valuable insights from large datasets.

6 References

- Riga Technical University, Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems, DSP201 – Database Management Systems, Presentations