**Лабораторная работа № 6. Группировка данных**

1. На основе таблицы **AUDITORIUM** разработать SELECT-запрос, вычисляющий максимальную, минимальную и среднюю вместимость аудиторий, суммарную вместимость всех аудиторий и общее количество аудиторий.

use UNIVER;

SELECT

MIN(AUDITORIUM.AUDITORIUM\_CAPACITY) [Min\_Capacity],

MAX(AUDITORIUM.AUDITORIUM\_CAPACITY) [Max\_Capacity],

AVG(AUDITORIUM.AUDITORIUM\_CAPACITY) [Average\_Capacity],

SUM(AUDITORIUM.AUDITORIUM\_CAPACITY) [Total\_Capacity],

COUNT(\*) [Auditorium\_Count]

FROM AUDITORIUM;



1. На основе таблиц **AUDITORIUM** и **AUDITORIUM\_TYPE** разработать запрос, вычисляющий для каждого типа аудиторий максимальную, минимальную, среднюю вместимость аудиторий, суммарную вместимость всех аудиторий и общее количество аудиторий данного типа. Результирующий набор должен содержать столбец с наименованием типа аудиторий (столбец **AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME**) и столбцы с вычисленными величинами. Использовать внутреннее соединение таблиц, секцию GROUP BY и агрегатные функции.

SELECT

AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME,

MAX(AUDITORIUM.AUDITORIUM\_CAPACITY) [Max\_Capacity],

MIN(AUDITORIUM.AUDITORIUM\_CAPACITY) [Min\_Capacity],

AVG(AUDITORIUM.AUDITORIUM\_CAPACITY) [Average\_Capacity],

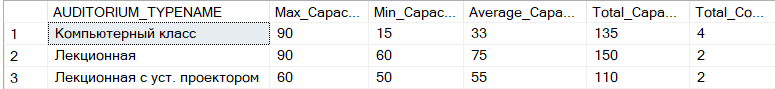
SUM(AUDITORIUM.AUDITORIUM\_CAPACITY) [Total\_Capacity],

COUNT(AUDITORIUM.AUDITORIUM\_TYPE) [Total\_Count]

FROM AUDITORIUM

JOIN AUDITORIUM\_TYPE ON AUDITORIUM.AUDITORIUM\_TYPE = AUDITORIUM\_TYPE.AUDITORIUM\_TYPE

GROUP BY AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME;



1. Разработать запрос на основе таблицы **PROGRESS**, который содержит количество экзаменационных оценок в заданном интервале. При этом учесть, что сортировка строк должна осуществляться в порядке, обратном величине оценки; сумма значений в столбце **количество** должна быть равна количеству строк в таблице **PROGRESS**. Использовать подзапрос в секции FROM, в подзапросе применить GROUP BY, сортировку осуществить во внешнем запросе. В секции GROUP BY, в SELECT-списке подзапроса и в ORDER BY внешнего запроса применить CASE.

SELECT \*

FROM (

SELECT

CASE

when (PROGRESS.NOTE = 10) then '10'

when (PROGRESS.NOTE IN (8, 9)) then '8-9'

when (PROGRESS.NOTE IN (6, 7)) then '6-7'

when (PROGRESS.NOTE IN (4, 5)) then '4-5'

END [Notes],

COUNT(\*) [Count]

FROM PROGRESS

GROUP BY

CASE

when (PROGRESS.NOTE = 10) then '10'

when (PROGRESS.NOTE IN (8, 9)) then '8-9'

when (PROGRESS.NOTE IN (6, 7)) then '6-7'

when (PROGRESS.NOTE IN (4, 5)) then '4-5'

END

) AS T

ORDER BY

CASE [Notes]

when '10' then 1

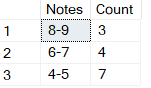
when '8-9' then 2

when '6-7' then 3

when '4-5' then 4

else 0

END;



1. Разработать SELECT-запроса на основе таблиц **FACULTY**, **GROUPS**, **STUDENT** и **PROGRESS**, который содержит среднюю экзаменационную оценку для каждого курса каждой специальности. Строки отсортировать в порядке убывания средней оценки. При этом следует учесть, что средняя оценка должна рассчитываться с точностью до двух знаков после запятой. Использовать внутреннее соединение таблиц, агрегатную функцию AVG и встроенные функции CAST и ROUND.

Переписать SELECT-запрос, разработанный в задании 4 так, чтобы в расчете среднего значения оценок использовались оценки только по дисциплинам с кодами **БД** и **ОАиП**. Использовать WHERE.

SELECT

FACULTY.FACULTY,

GROUPS.PROFESSION,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM FACULTY

JOIN GROUPS ON FACULTY.FACULTY = GROUPS.FACULTY

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

GROUP BY

FACULTY.FACULTY,

GROUPS.PROFESSION

ORDER BY [Average\_Note];

SELECT

FACULTY.FACULTY,

GROUPS.PROFESSION,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM FACULTY

JOIN GROUPS ON FACULTY.FACULTY = GROUPS.FACULTY

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

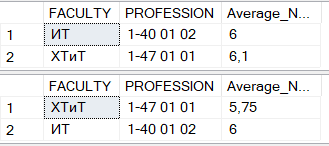
WHERE PROGRESS.SUBJECT\_T IN ('ОАиП', 'СУБД')

GROUP BY

FACULTY.FACULTY,

GROUPS.PROFESSION

ORDER BY [Average\_Note];



1. На основе таблиц **FACULTY**, **GROUPS**, **STUDENT** и **PROGRESS** разработать SELECT-запрос, в котором выводятся специальность, дисциплины и средние оценки при сдаче экзаменов на факультете ТОВ. Использовать группировку по полям FACULTY, PROFESSION, SUBJECT. Добавить в запрос конструкцию **ROLLUP** и проанализировать результат.

SELECT

FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM FACULTY

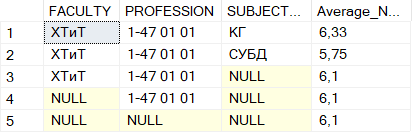
JOIN GROUPS ON FACULTY.FACULTY = GROUPS.FACULTY

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE FACULTY.FACULTY = 'ХТиТ'

GROUP BY ROLLUP(GROUPS.PROFESSION, FACULTY.FACULTY, PROGRESS.SUBJECT\_T);



1. Выполнить исходный SELECT-запрос п.5 с использованием **CUBE**-группировки. Проанализировать результат.

SELECT

FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM FACULTY

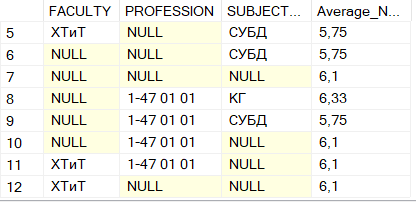
JOIN GROUPS ON FACULTY.FACULTY = GROUPS.FACULTY

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE FACULTY.FACULTY = 'ХТиТ'

GROUP BY CUBE(GROUPS.PROFESSION, FACULTY.FACULTY, PROGRESS.SUBJECT\_T);



1. На основе таблиц **GROUPS**, **STUDENT** и **PROGRESS** разработать SELECT-запрос, в котором определяются результаты сдачи экзаменов. В запросе должны отражаться специальности, дисциплины, средние оценки студентов на факультете ТОВ. Отдельно разработать запрос, в котором определяются результаты сдачи экзаменов на факультете ХТиТ. Объединить результаты двух запросов с использованием операторов UNION и UNION ALL. Объяснить результаты.

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON STUDENT.IDGROUP = GROUPS.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ИТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

UNION

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON STUDENT.IDGROUP = GROUPS.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ХТиТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

ORDER BY [Average\_Note] DESC;

-- UNION ALL

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON STUDENT.IDGROUP = GROUPS.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ИТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

UNION ALL

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

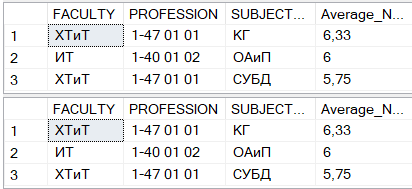
JOIN STUDENT ON STUDENT.IDGROUP = GROUPS.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ХТиТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

ORDER BY [Average\_Note] DESC;



1. Получить пересечение двух множеств строк, созданных в результате выполнения запросов пункта 7. Объяснить результат. Использовать оператор INTERSECT.

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ИТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

INTERSECT

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ХТиТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

ORDER BY [Average\_Note] DESC;



1. Получить разницу между множеством строк, созданных в результате запросов пункта 8. Объяснить результат. Использовать оператор EXCEPT.

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ИТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

INTERSECT

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ХТиТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

EXCEPT

SELECT

GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT\_T,

ROUND(AVG(CAST(PROGRESS.NOTE AS float(2))), 2) [Average\_Note]

FROM GROUPS

JOIN STUDENT ON GROUPS.IDGROUP = STUDENT.IDGROUP

JOIN PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

WHERE GROUPS.FACULTY = 'ИТ'

GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT\_T

ORDER BY [Average\_Note] DESC;



1. На основе таблицы **PROGRESS** определить для каждой дисциплины количество студентов, получивших оценки 8 и 9. Использовать группировку, секцию HAVING, сортировку.

SELECT DISTINCT

a.SUBJECT\_T,

(

SELECT COUNT(\*)

FROM PROGRESS b

WHERE a.SUBJECT\_T = b.SUBJECT\_T

AND b.NOTE IN (8, 9)

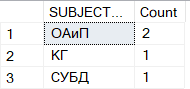
) [Count]

FROM PROGRESS a

GROUP BY a.SUBJECT\_T, a.NOTE

HAVING a.NOTE IN (8, 9)

ORDER BY [Count] DESC;



11. Разработать и выполнить аналогичные запросы для базы данных **X\_MyBASE**.

use Dobriyan\_MyBase;

-- #1

SELECT

MIN(Groups.Students\_Count) [Min\_Capacity],

MAX(Groups.Students\_Count) [Max\_Capacity],

AVG(Groups.Students\_Count) [Average\_Capacity],

SUM(Groups.Students\_Count) [Total\_Capacity],

COUNT(\*) [Groups\_Count]

FROM Groups;

-- #2

SELECT

Specialities.Speciality,

MIN(Groups.Students\_Count) [Min\_Capacity],

MAX(Groups.Students\_Count) [Max\_Capacity],

AVG(Groups.Students\_Count) [Average\_Capacity],

SUM(Groups.Students\_Count) [Total\_Capacity],

COUNT(Groups.Speciality) [Groups\_Count]

FROM Groups

JOIN Specialities ON Groups.Speciality = Specialities.id

GROUP BY Specialities.Speciality;

-- #3

SELECT \*

FROM (

SELECT

CASE

when (Courses.Cost > 300) then '> 300'

when (Courses.Cost BETWEEN 271 AND 300) then '271-300'

when (Courses.Cost BETWEEN 250 AND 270) then '250-270'

else '< 250'

END [Courses\_Cost],

COUNT(\*) [Count]

FROM Courses

GROUP BY

CASE

when (Courses.Cost > 300) then '> 300'

when (Courses.Cost BETWEEN 271 AND 300) then '271-300'

when (Courses.Cost BETWEEN 250 AND 270) then '250-270'

else '< 250'

END

) AS T

ORDER BY

CASE [Courses\_Cost]

when '> 300' then 1

when '271-300' then 2

when '250-270' then 3

else 0

END;

-- #4

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

GROUP BY Specialities.Speciality

ORDER BY [Avg\_Students\_Count] DESC;

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality IN ('ПОИТ', 'ДЭиВИ')

GROUP BY Specialities.Speciality

ORDER BY [Avg\_Students\_Count] DESC;

-- #5

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality IN ('ПОИТ', 'ДЭиВИ')

GROUP BY ROLLUP(Specialities.Speciality)

ORDER BY [Avg\_Students\_Count] DESC;

-- #6

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality IN ('ПОИТ', 'ДЭиВИ')

GROUP BY CUBE(Specialities.Speciality)

ORDER BY [Avg\_Students\_Count] DESC;

-- #7

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ПОИТ'

GROUP BY CUBE(Specialities.Speciality)

UNION

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ДЭиВИ'

GROUP BY CUBE(Specialities.Speciality)

ORDER BY [Avg\_Students\_Count] DESC;

-- UNION ALL

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ПОИТ'

GROUP BY CUBE(Specialities.Speciality)

UNION ALL

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ДЭиВИ'

GROUP BY CUBE(Specialities.Speciality)

ORDER BY [Avg\_Students\_Count] DESC;

-- #8

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ПОИТ'

GROUP BY CUBE(Specialities.Speciality)

INTERSECT

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ДЭиВИ'

GROUP BY CUBE(Specialities.Speciality)

ORDER BY [Avg\_Students\_Count] DESC;

-- #9

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ПОИТ'

GROUP BY CUBE(Specialities.Speciality)

INTERSECT

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ДЭиВИ'

GROUP BY CUBE(Specialities.Speciality)

EXCEPT

SELECT

Specialities.Speciality,

COUNT(Groups.Number) [Groups\_Count],

ROUND(AVG(CAST(Groups.Students\_Count AS float(2))), 2) [Avg\_Students\_Count]

FROM Specialities

JOIN Groups ON Specialities.id= Groups.Speciality

WHERE Specialities.Speciality = 'ПОИТ'

GROUP BY CUBE(Specialities.Speciality)

ORDER BY [Avg\_Students\_Count] DESC;

-- #10

SELECT DISTINCT

Specialities.Speciality,

(

SELECT COUNT(\*)

FROM Groups b

WHERE a.Speciality = b.Speciality

AND b.Students\_Count BETWEEN 15 AND 20

) [Groups with 15-20 students]

FROM Groups a

JOIN Specialities ON a.Speciality = Specialities.id

GROUP BY a.Students\_Count, a.Speciality, Specialities.Speciality

HAVING a.Students\_Count BETWEEN 15 AND 20

ORDER BY [Groups with 15-20 students] DESC;