**Code Space, SQL exercises**

Table of content

[Activity 1 – Introductory SQL 2](#_Toc147342098)

[Exercise01 2](#_Toc147342099)

[Exercise02 3](#_Toc147342100)

[Exercise03 3](#_Toc147342101)

[Exercise04 4](#_Toc147342102)

[Activity 2 - Querying Data and Sorting Data 4](#_Toc147342103)

[Exercise01 4](#_Toc147342104)

[Exercise02 5](#_Toc147342105)

[Exercise03 5](#_Toc147342106)

[Exercise04 5](#_Toc147342107)

[Activity 3 - Filtering Data Part01 6](#_Toc147342108)

[Exercise01 6](#_Toc147342109)

[Exercise02 6](#_Toc147342110)

[Exercise03 6](#_Toc147342111)

[Exercise04 7](#_Toc147342112)

[Exercise05 7](#_Toc147342113)

[Exercise06 8](#_Toc147342114)

[Activity 4 - Filtering Data Part02 8](#_Toc147342115)

[Exercise01 8](#_Toc147342116)

[Exercise02 8](#_Toc147342117)

[Exercise03 9](#_Toc147342118)

[Exercise04 9](#_Toc147342119)

[Exercise05 10](#_Toc147342120)

[Exercise06 10](#_Toc147342121)

[Exercise07 10](#_Toc147342122)

[Exercise08 11](#_Toc147342123)

[Activity 5 - Joins 11](#_Toc147342124)

[Exercise01 11](#_Toc147342125)

[Exercise02 12](#_Toc147342126)

[Exercise03 13](#_Toc147342127)

[Activity 6 - Aggregate Functions 13](#_Toc147342128)

[Exercise01 13](#_Toc147342129)

[Exercise02 14](#_Toc147342130)

[Exercise03 14](#_Toc147342131)

[Exercise04 15](#_Toc147342132)

[Exercise05 15](#_Toc147342133)

[Exercise06 16](#_Toc147342134)

[Activity 7 - Grouping Data 16](#_Toc147342135)

[Exercise01 16](#_Toc147342136)

[Exercise02 16](#_Toc147342137)

[Exercise03 17](#_Toc147342138)

[Activity 8 - Subquery Part01 17](#_Toc147342139)

[Exercise01 17](#_Toc147342140)

[Exercise02 18](#_Toc147342141)

[Exercise03 18](#_Toc147342142)

[Exercise04 19](#_Toc147342143)

[Exercise05 19](#_Toc147342144)

[Activity 9 - Subquery Part02 20](#_Toc147342145)

[Exercise01 20](#_Toc147342146)

[Exercise02 21](#_Toc147342147)

[Activity 10 - Modifying Data 21](#_Toc147342148)

[Exercise01 21](#_Toc147342149)

[Exercise02 21](#_Toc147342150)

[Exercise03 22](#_Toc147342151)

[Exercise04 22](#_Toc147342152)

[Exercise05 23](#_Toc147342153)

## Activity 1 – Introductory SQL

Exercise01: Complete the SQL and copy the SQL following by screenshots of the output (if the output is too large – only do partial screenshot and mention how many records in total is the output).

Code:



Output:

A screenshot of a computer

Description automatically generated

Number of records: 31

Exercise02: Do the same as the above, this time using the other three tables provided, choose any three attributes and then display these in the output. Show the SQL code and screenshot the output (as usual). Use of the DISTINCT command after the SELECT statement. Applying the DISTINCT command across the primary key doesn’t have any purpose, since all the records would naturally expected to appear. Since the primary key by its very definition is that the values need to be unique. e.g. SELECT DISTINCT PatientID FROM tblpatient; A more purposeful application of DISTINCT would be showing the counties that the patients come from.

Code:

SELECT DISTINCT `LastName` FROM `tbldoctors`;

Output:

A screenshot of a computer program

Description automatically generated

Code:

SELECT DISTINCT `County` FROM `tblpatient

Output:

A screenshot of a computer

Description automatically generated

Code:

SELECT DISTINCT `PostCode` FROM `tblreceptionist`

Output:

A screenshot of a computer

Description automatically generated

Exercise03: Apply the DISTINCT command to identify the counties that the patients come from using the patients table. This would give an understanding of the geographic spread.

Code:

SELECT `County` FROM `tblpatient`

Output:

A screenshot of a computer program

Description automatically generated

Number of records: 25

Exercise04: Also apply the DISTINCT on the title attribute, which would suggest to us, what age groups the patients are being drawn from.

Code:

SELECT DISTINCT `DOB` FROM `tblpatient`

Output:

A screenshot of a computer program

Description automatically generated

Number of records: 25

## Activity 2 - Querying Data and Sorting Data

Exercise01: Using locations select \* attributes and display. Then use Select for three attributes including the primary key and display. Make sure to distinguish where the primary key is coming from in terms of the table name.

Code:

SELECT job\_id, job\_title, min\_salary FROM jobs;

Output:

A screenshot of a computer

Description automatically generated

Exercise02: Create an sql – that will show the employee\_id, last name and their salary. Then use an alias for the table name – simply use the letter e to reference it. Instead of salary use the alias ‘Monthly Salary’ for the last attribute.

Code:

SELECT employee\_id, last\_name, salary / 12,0 AS 'Monthly salary'FROM employees;

Output:

A screenshot of a computer

Description automatically generated

Exercise03: On the select clause – combine the persons first name with their name last to be shown as the alias called ‘Fullname’.

Code:

SELECT (first\_name||' '||last\_name) AS 'Fulname' FROM employees

ORDER BY last\_name DESC;

Output:

A screenshot of a computer

Description automatically generated

Exercise04: Complete an sql that will display a person’s lastname, email, salary and have the sort done according to the person salary in descending order (so highest will be at the top), if two people have the same salary then the secondary sort should be lastname in ascending order.

Code:

SELECT last\_name, email, salary FROM employees

ORDER BY salary DESC, last\_name ASC;

Output:

A screenshot of a computer

Description automatically generated

## Activity 3 - Filtering Data Part01

Exercise01: Apply the distinct command to determine the individual department ids from the employees table. This should give a straight listing of only the departments id’s without duplication.

Code:

SELECT DISTINCT department\_id FROM employees;

Output:

A screenshot of a computer

Description automatically generated

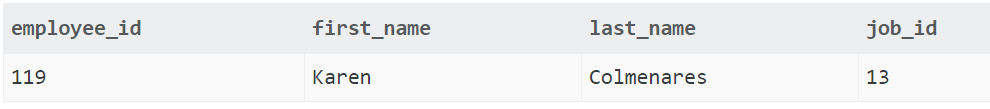
The number of records: 11

Exercise02: Determine the top salary within the organisation. Using the employees table. Display their employee\_id, first name, last name, job\_id.

Code:

SELECT employee\_id, first\_name, last\_name, job\_id FROM employees WHERE salary>1000 ORDER BY salary ASC LIMIT 1

Output:



Exercise03: Display a listing of the department names, and department id’s – starting with the 5th department and continuing the 10th department id. Assuming an ascending sort is done on department id.

Code:

SELECT department\_id, department\_name FROM departments

WHERE department\_id >= 5 AND department\_id <= 10

ORDER BY department\_id ASC;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 6

Exercise04: Complete a distinct command on the location id’s within the department table. To show only a listing of the unique location id’s that the company has.

Code:

SELECT DISTINCT location\_id FROM departments;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 7

Exercise05: Determine the details – first name, last name, salary, phone number of those employees earning under or equal to 5000.

Code:

SELECT first\_name, last\_name, salary, phone\_number

FROM employees

WHERE salary <= 5000;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 12

Exercise06: Determine employee\_id, first name, last name , hire date to show those people hired after the beginning on 1999.

Code:

SELECT employee\_id, first\_name, last\_name, hire\_date

FROM employees

WHERE hire\_date >= 1999

ORDER BY hire\_date DESC;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 5

## Activity 4 - Filtering Data Part02

Exercise01: Write the sql code that will determine anyone whose name begins with ‘P’ and whose salary is below 8000.

Code:

SELECT \* FROM employees

WHERE last\_name LIKE 'P%'

AND salary < 8000;

Output:

A screenshot of a computer

Description automatically generated

Exercise02: Identify the employees hired in 1997, 1998. They all have a department id of 5.

Code:

SELECT \* FROM employees

WHERE hire\_date BETWEEN 1997 AND 1998

AND department\_id IN (5);

Output:

A screenshot of a computer

Description automatically generated

Exercise03: Identify employees that are employed in 1998, 1999, 2000. Display their first name, last name, salary, email, hire\_date (alias ‘Year of Hire’) and order by last name in ascending order.

Code:

SELECT first\_name, last\_name, salary, email, hire\_date AS 'Year of Hire'

FROM employees

WHERE hire\_date BETWEEN 1998 AND 2000

ORDER BY last\_name ASC;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 10

Exercise04: Display all the details of the employees that have a salary that is bounded between 5000 and 10000. (inclusive). Use the inequalities to generate the output within the where statement (>, <, =).

Code:

SELECT \* FROM employees

WHERE salary >= 5000 AND salary <= 10000;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 19

Exercise05: Using the between operator to determine the salary – determine the employees that have salaries between 4000 and 12000. In ascending order according to first name.

Code:

SELECT \* FROM employees

WHERE salary BETWEEN 4000 AND 12000

ORDER BY first\_name ASC;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 28

Exercise06: Display all the records for employees that are NOT in the following departments – they are not in 1, 4, 8, 10. So it will display all the others. Do this with a sort according to salary in descending order.

Code:

SELECT \* FROM employees

WHERE department\_id <> 1 AND department\_id <> 4 AND department\_id <> 8 AND department\_id <> 10

ORDER BY salary DESC;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 26

Exercise07: Display the details of the employees that have an ‘s’ at the end of their first name. Sort using ascending sort according to first name.

Code:

SELECT \* FROM employees

WHERE first\_name LIKE '%s'

ORDER BY first\_name ASC;

Output:

A screenshot of a computer

Description automatically generated

Exercise08: Display the details of employees that have three letters unknown followed by a ‘g’ for their last name.

Code:

SELECT \* FROM employees

WHERE last\_name LIKE '\_\_\_g%';

Output:

A screenshot of a computer

Description automatically generated

## Activity 5 - Joins

Exercise01: Create an sql that will display the following fields (attributes) – first name, last name, manager\_id, department\_id, location\_id. Using an EQUI join display the data with an appropriate query.

Code:

SELECT

first\_name,

last\_name,

manager\_id,

employees.department\_id,

location\_id

FROM

employees

INNER JOIN

departments ON departments.department\_id = employees.department\_id

Output:

A screenshot of a computer

Description automatically generated

The number of records: 40

Exercise02: Create an sql that will display the following fields (attributes) – location\_id, postal\_code, country\_id, country\_name – apply a series of alias’ to the relevant tables. Then generate the output according to an INNER JOIN. With the primary key ‘location\_id’ deciding what will be the prevalent table in the query.

Code:

SELECT

location\_id,

postal\_code,

l.country\_id,

country\_name

FROM

locations l

INNER JOIN

countries c ON c.country\_id = l.country\_id;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 7

Exercise03: Create a query that will generate the following information, employee\_id, first\_name, last\_name, dependent\_id, first\_name, last\_name, relationship. This will draw information from the employees and dependents tables.

Code:

SELECT

e.employee\_id,

e.first\_name,

e.last\_name,

d.first\_name,

d.last\_name,

relationship

FROM employees e

INNER JOIN

dependents d ON d.employee\_id = e.employee\_id;

output:

A screenshot of a computer

Description automatically generated

The number of records: 30

## Activity 6 - Aggregate Functions

Exercise01: Determine the number of employees in the Sales department. Also create a proper header that shows the information appropriately.

Code:

SELECT SUM(department\_id = 8) AS 'Sales department'

FROM employees;

Output:

A close-up of a text

Description automatically generated

Exercise02: Determine the number of employing working in each of the respective departments.

Code:

SELECT

SUM(department\_id = 1) AS 'Number employees in administration department',

SUM(department\_id = 2) AS 'Number employees in marketing department',

SUM(department\_id = 3) AS 'Number employees in purchasing department',

SUM(department\_id = 4) AS 'Number employees in human resources department',

SUM(department\_id = 5) AS 'Number employees in shipping department',

SUM(department\_id = 6) AS 'Number employees in IT department',

SUM(department\_id = 7) AS 'Number employees in public relations department',

SUM(department\_id = 8) AS 'Number employees in sales department',

SUM(department\_id = 9) AS 'Number employees in executive department',

SUM(department\_id = 10) AS 'Number employees in finance department',

SUM(department\_id = 11) AS 'Number employees in accounting department'

FROM employees;

Output:

A close-up of a message

Description automatically generated

The number of records: 11

Exercise03: Determine the number of employees each manager will manage from the employees table.

Code:

SELECT

SUM(manager\_id = 100) AS 'manager ID 100',

SUM(manager\_id = 101) AS 'manager ID 101',

SUM(manager\_id = 102) AS 'manager ID 102',

SUM(manager\_id = 103) AS 'manager ID 103',

SUM(manager\_id = 108) AS 'manager ID 108',

SUM(manager\_id = 114) AS 'manager ID 114',

SUM(manager\_id = 120) AS 'manager ID 120',

SUM(manager\_id = 123) AS 'manager ID 123',

SUM(manager\_id = 201) AS 'manager ID 201',

SUM(manager\_id = 205) AS 'manager ID 205',

SUM(manager\_id = null) AS 'manager ID unlnown'

FROM employees;

Output:

A screenshot of a computer

Description automatically generated

The number of records: 11

Exercise04: Determine who (if anyone) has two children from the dependents table.

Code:

SELECT last\_name AS 'Children last names', employee\_id AS 'Emploee ID who has two children'

FROM dependents

GROUP BY employee\_id

HAVING COUNT(employee\_id) = 2;

Output:

There is not employees having two children.

Exercise05: Determine the amount of salary per department. Give a breakdown using only the department\_id’s in the output along with the salary amounts.

Code:

SELECT department\_id, SUM(salary) FROM employees

GROUP BY department\_id;

Output:

A screenshot of a computer screen

Description automatically generated

The number of records: 11

Exercise06: Determine the average of the max salary inside of the jobs table. Round to two decimal places.

Code:

SELECT ROUND(AVG(max\_salary), 2) AS 'average of the max salary ' FROM jobs;

Output:

A screenshot of a computer

Description automatically generated

## Activity 7 - Grouping Data

Exercise01: Create a group by operation on job\_id that will show the number of counts that each job\_id corresponds to.

Code:

SELECT COUNT(job\_id) AS 'Operations on job ID', department\_id

FROM employees

GROUP BY department\_id

HAVING SUM(job\_id);

Output:

A screenshot of a computer

Description automatically generated

The number of records: 11

Exercise02: Then do a where operation on the same series, that will display the output from

job\_id = 9.

Code:

SELECT COUNT(job\_id) AS 'Operations on job ID', department\_id

FROM employees

GROUP BY department\_id

HAVING SUM(job\_id = 9);

Output:

A close-up of a white rectangular object

Description automatically generated

Exercise03: Determine the job titles alongside of the job\_ids for the original query.

Code:

SELECT

COUNT(e.job\_id) AS 'Jobs at the position',

j.job\_id,

j.job\_title

FROM employees e

INNER JOIN

jobs j ON j.job\_id = e.job\_id

GROUP BY j.job\_id

HAVING SUM(e.job\_id);

Output:

A white grid with black text

Description automatically generated

The number of records: 19

## Activity 8 - Subquery Part01

Exercise01: Identify the person at the low end of the salary scale – using a subquery to identify the amount they receive. Then use this amount to work out their employee details.

Code:

SELECT employee\_id, first\_name, last\_name, salary

FROM employees

WHERE

salary =

(SELECT MIN(salary)

FROM employees)

ORDER BY employee\_id, first\_name, last\_name;

Output:

A close-up of a computer screen

Description automatically generated

Exercise02: Write the sql code to determine the employees (employee\_id, first\_name, last\_name, salary) for those earning above the average salary.

Code:

SELECT employee\_id, first\_name, last\_name, salary

FROM employees

WHERE

salary >

(SELECT AVG(salary)

FROM employees)

ORDER BY employee\_id, first\_name, last\_name;

Output:

A screenshot of a computer

Description automatically generated

Exercise03: Determine the people earning below the average inside of department\_id = 6. Then show their details (employee\_id, first\_name, last\_name, salary, department\_id).

Code:

SELECT employee\_id, first\_name, last\_name, salary, department\_id

FROM employees

WHERE

salary <

(SELECT AVG(salary)

FROM employees

WHERE department\_id = 6)

ORDER BY employee\_id, first\_name, last\_name;

Output:

A screenshot of a computer

Description automatically generated

Exercise04: Determine the employees working in the IT department for the company.

Code:

SELECT \*

FROM employees

WHERE

department\_id = 6

ORDER BY employee\_id, first\_name, last\_name;

Output:

A screenshot of a computer

Description automatically generated

Number of records: 5

Exercise05: Starting with ‘Jennifer King’ determine the employees name i.e. the father to this child. Who is it?

Code:

SELECT

e.employee\_id,

e.first\_name,

e.last\_name,

d.dependent\_id,

d.first\_name,

d.last\_name

FROM employees e

INNER JOIN dependents d ON d.employee\_id = e.employee\_id

WHERE EXISTS

(SELECT \*

FROM dependents d

WHERE

d.employee\_id = e.employee\_id AND d.dependent\_id > 3)

ORDER BY d.dependent\_id, e.first\_name, e.last\_name;

Output:

A screenshot of a computer

Description automatically generated

Number of records: 27

## Activity 9 - Subquery Part02

Exercise01: ALL clasule.

Code:

SELECT first\_name,

last\_name,

salary,

department\_id

FROM employees

WHERE salary =

(SELECT MAX(salary)

FROM employees

WHERE department\_id = 2)

ORDER BY salary;

Output:

A screenshot of a computer

Description automatically generated

Exercise02: ANY(IN) clasule.

Code:

SELECT first\_name,

last\_name,

salary,

department\_id

FROM employees

WHERE salary IN

(SELECT AVG(salary)

FROM employees

GROUP BY department\_id)

Output:

A screenshot of a computer

Description automatically generated

Number of records: 5

## Activity 10 - Modifying Data

Exercise01: Use the insert command to enter a record into the student table.

Code:

INSERT INTO `student`(`studentno`, `surname`, `firstname`, `email`, `mobileno`) VALUES ('EC007','Flinstone','Fred','flinstone@bt.com',1234567);

Output:

A screenshot of a computer

Description automatically generated

Exercise02: Enter the records for the other members of the Flintstone family: Wilma, Pebbles and Dino.

Code:

INSERT INTO `student`(`studentno`, `surname`, `firstname`, `email`, `mobileno`) VALUES

('EC008','Flinstone','Pebles','p.flinstone@bt.com',2345678),

('EC009','Flinstone','Wilma','w.flinstone@bt.com',3456789),

('EC010','Flinstone','Dino','d.flinstone@bt.com',4567891);

Output:

A screenshot of a computer

Description automatically generated

Exercise03: Insert into this table the following record.

Code1:

ALTER TABLE `student` ADD moduleno varchar(10), ADD modulename varchar(10), ADD moduleunitsize varchar(10), ADD hoursfordelivery varchar(10);

Code2:

UPDATE`student`

SET

Moduleno = 'DH3J34',

Modulename = 'SQL: Introduction',

Moduleunitsize = '1',

Hoursfordelivery = '32'

WHERE studentno = 'EC007';

Output:

A screenshot of a computer

Description automatically generated

Exercise04: Use code to alter the value from Fred to Barney.

Code:

UPDATE`student`

SET

firstname = 'Barney'

WHERE `mobileno` = '1234567';

Output:



Exercise05: Do sql that will show Barney and the module details – using a join.

Code:

SELECT surname, firstname, s.studentno, m.modulename

From

student s

INNER JOIN

module m ON m.studentno = s.studentno

Where

mobileno = '1234567';

Output:

A screenshot of a computer

Description automatically generated