Nama : Raden Sadiah Maharani

NIM : 2107126368

Basis Data Lanjut

Tugas Section 11

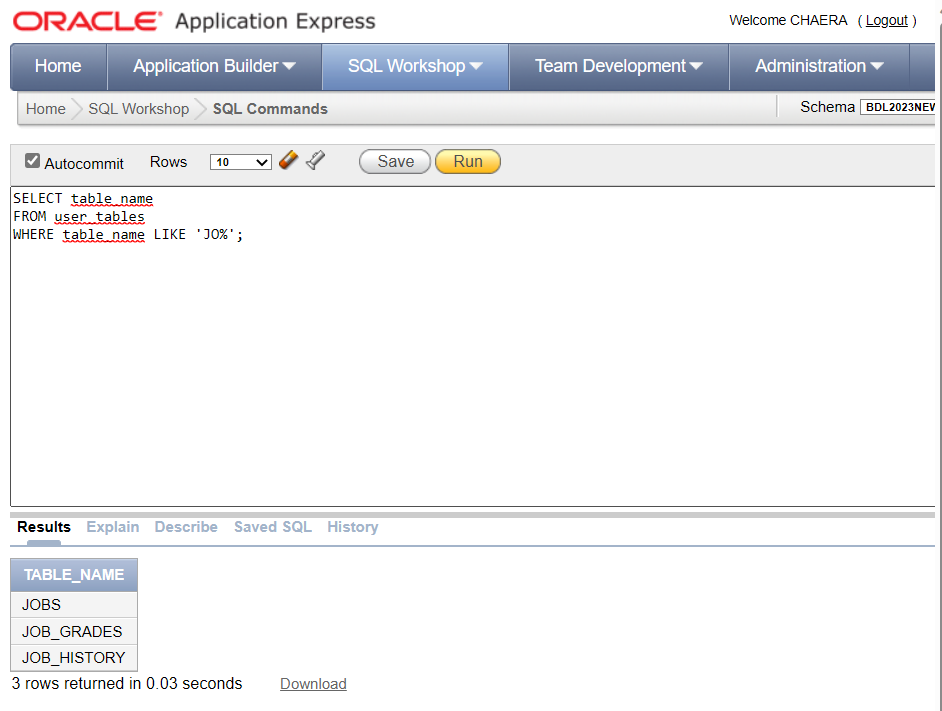
1. Create a list of all tables whose first two characters in the name of the table is JO. The tables must be owned by the current Oracle User.

→ SELECT table\_name

FROM user\_tables

WHERE table\_name LIKE 'JO%';

Result :

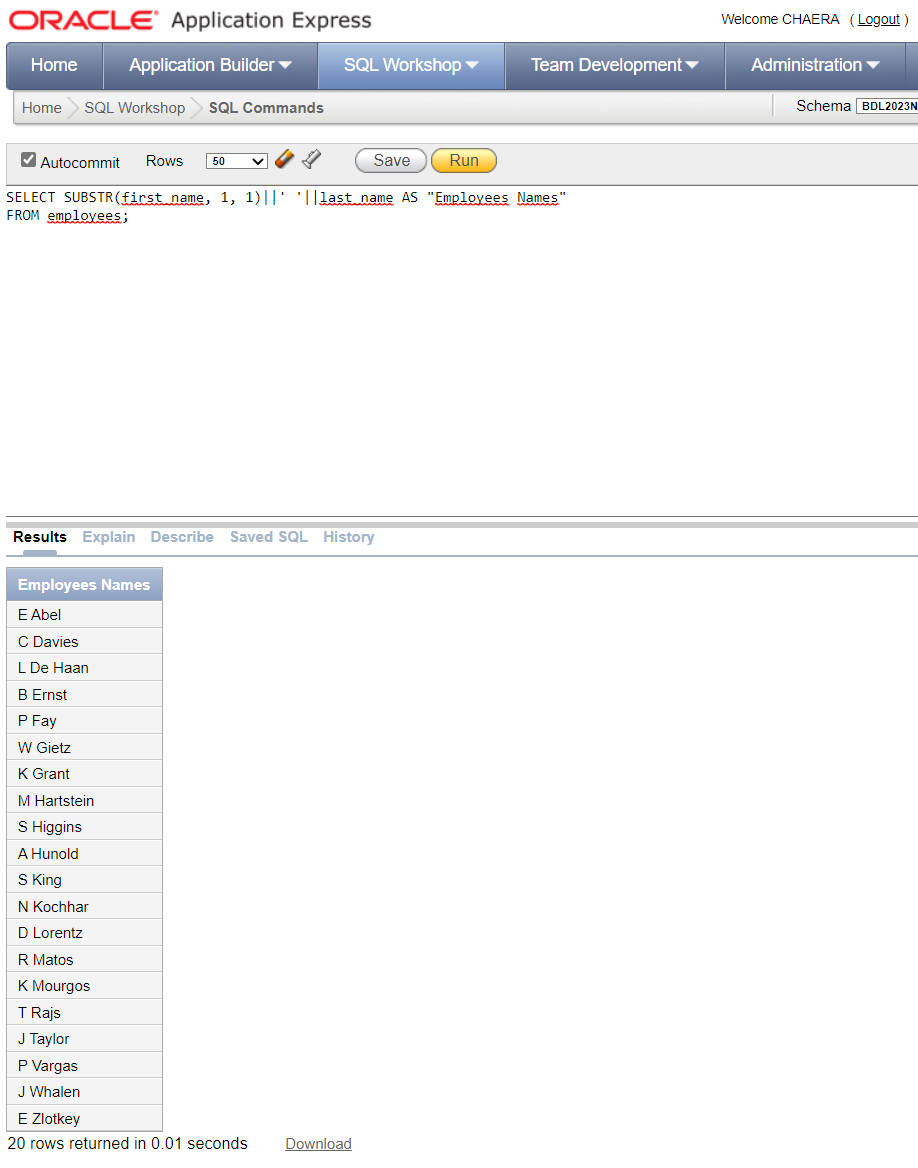


1. Create a list that includes the first initial of every employee's first name, a space, and the last name of the employee.

→ SELECT SUBSTR(first\_name, 1, 1)||' '||last\_name AS "Employees Names"

FROM employees;

Result :



1. Create a list of every employee's first name concatenated to a space and the employee's last name, and the email of all employees where the email address contains the string 'IN'.

→ SELECT first\_name||''||last\_name

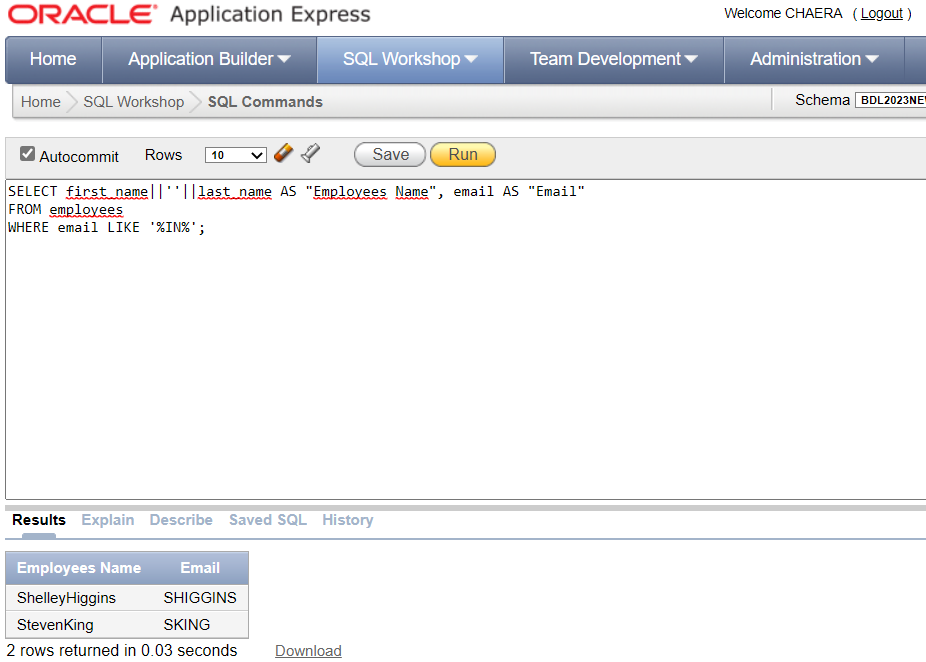
AS "Employees Name",

email AS "Email"

FROM employees

WHERE email LIKE '%IN%';

Result :



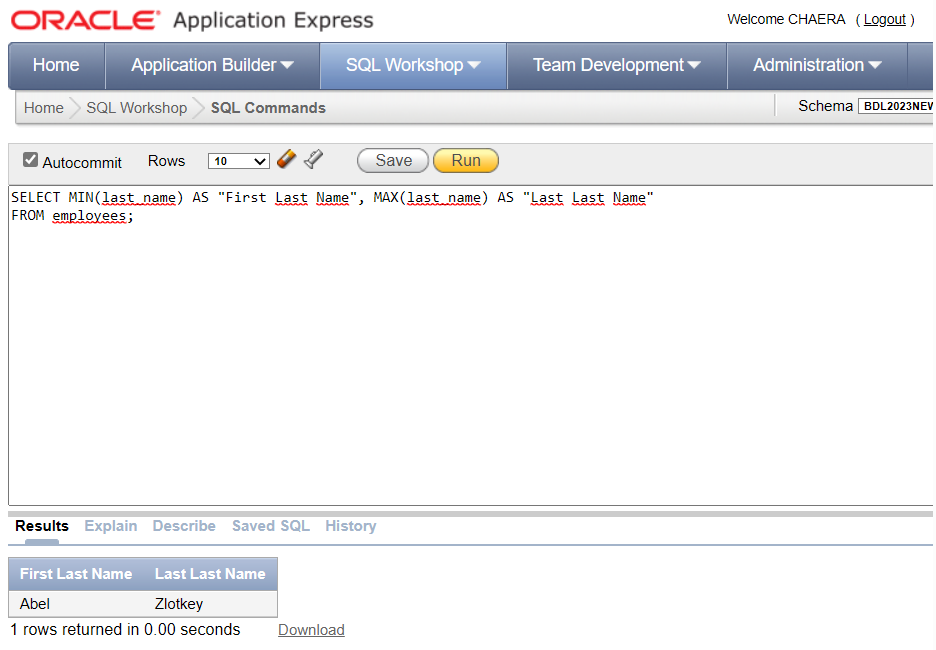
1. Create a list of 'smallest' last name and the 'highest' last name from the employees table.

→ SELECT MIN(last\_name) AS "First Last Name",

MAX(last\_name) AS "Last Last Name"

FROM employees;

Result :



1. Create a list of weekly salaries from the employees table where the weekly salary is between 700 and 3000. The salaries should be formatted to include a $- sign and have two decimal points like: $9999.99.

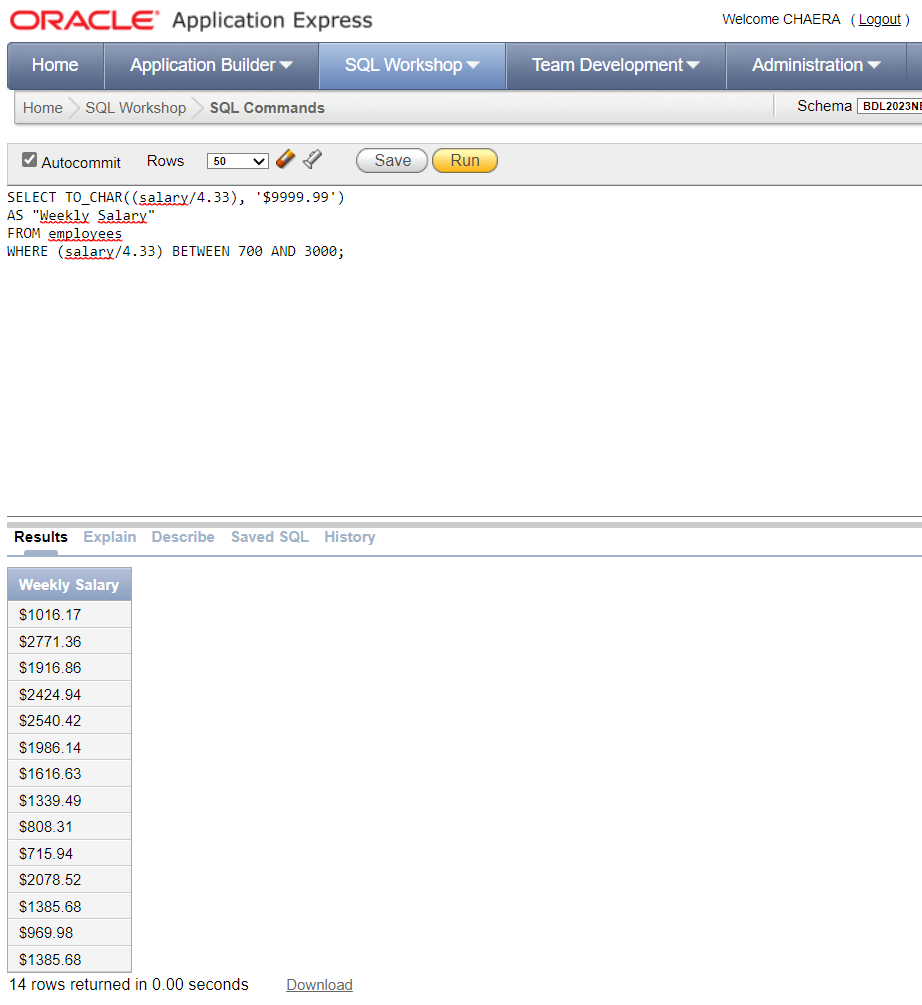
→ SELECT TO\_CHAR((salary/4.33), '$9999.99')

AS "Weekly Salary"

FROM employees

WHERE (salary/4.33) BETWEEN 700 AND 3000;

Result :



1. Create a list of every employee and his related job title sorted by job\_title.

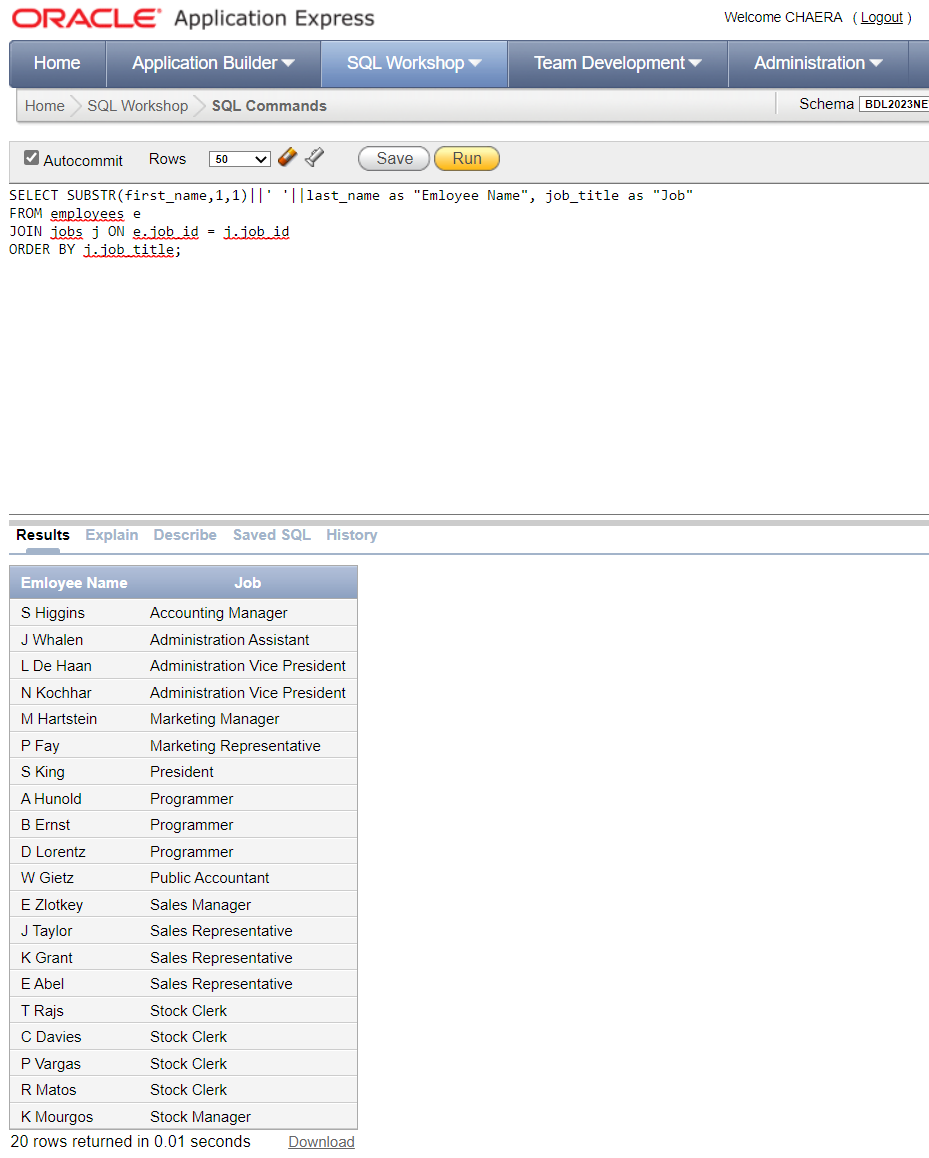
→ SELECT SUBSTR(first\_name,1,1)||' '||last\_name as "Emloyee Name", job\_title as "Job"

FROM employees e

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

ORDER BY j.job\_title;

Result :



1. Create a list of every employee’s job, the salary ranges within the job, and the employee's salary. List the lowest and highest salary range within each job with a dash to separate the salaries like this: 100 – 200.

→ SELECT SUBSTR(first\_name,1,1)||' '||last\_name as "Emloyee Name",

job\_title as "Job",

MIN(j.min\_salary) || ' - ' || MAX(j.max\_salary) as "Salary Range",

e.salary as "Employee's Salary"

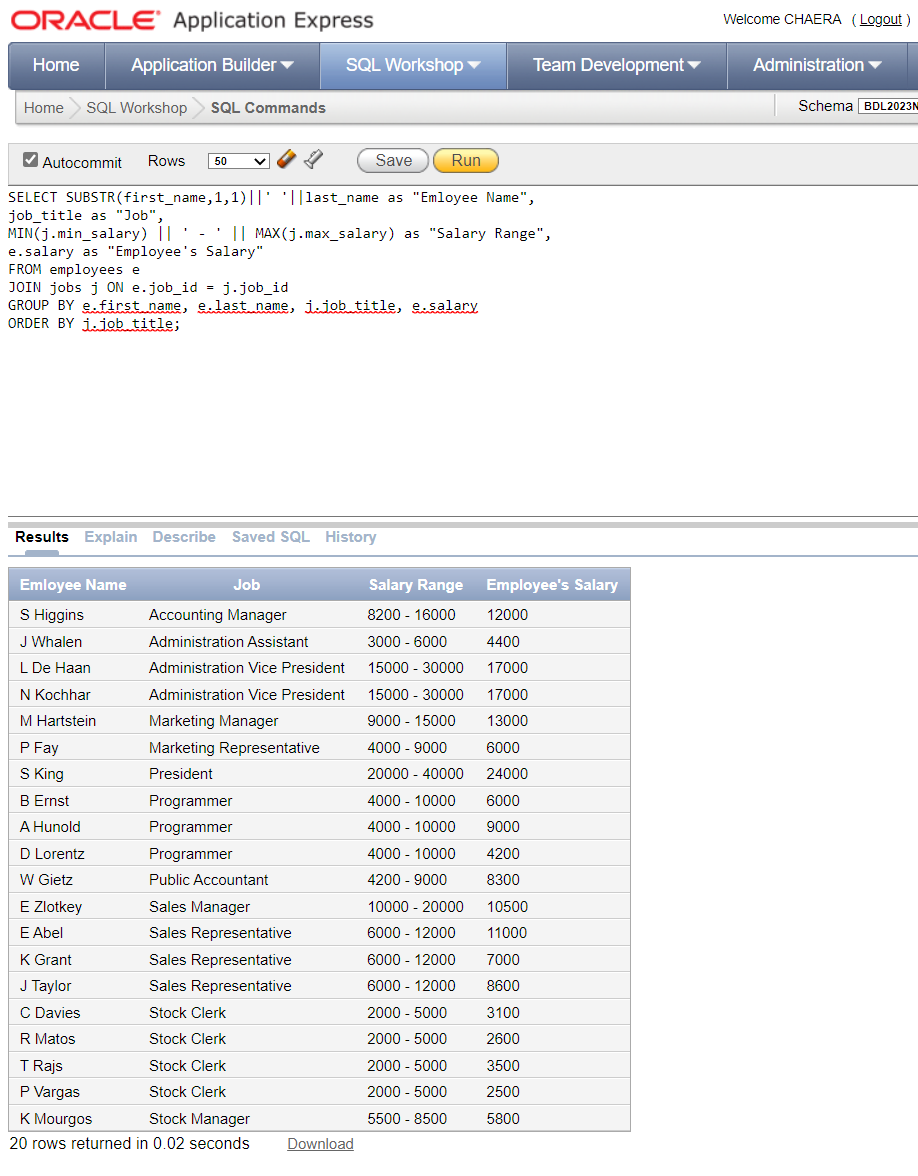
FROM employees e

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

GROUP BY e.first\_name, e.last\_name, j.job\_title, e.salary

ORDER BY j.job\_title;

Result :



1. Using an ANSII join method, create a list of every employee's first initial and last name, and department name. Make sure the tables are joined on all of the foreign keys declared between the two tables.

→ SELECT SUBSTR(first\_name, 1, 1) || ' ' || last\_name as "Employee Name", d.department\_name as "Department Name"

FROM employees e

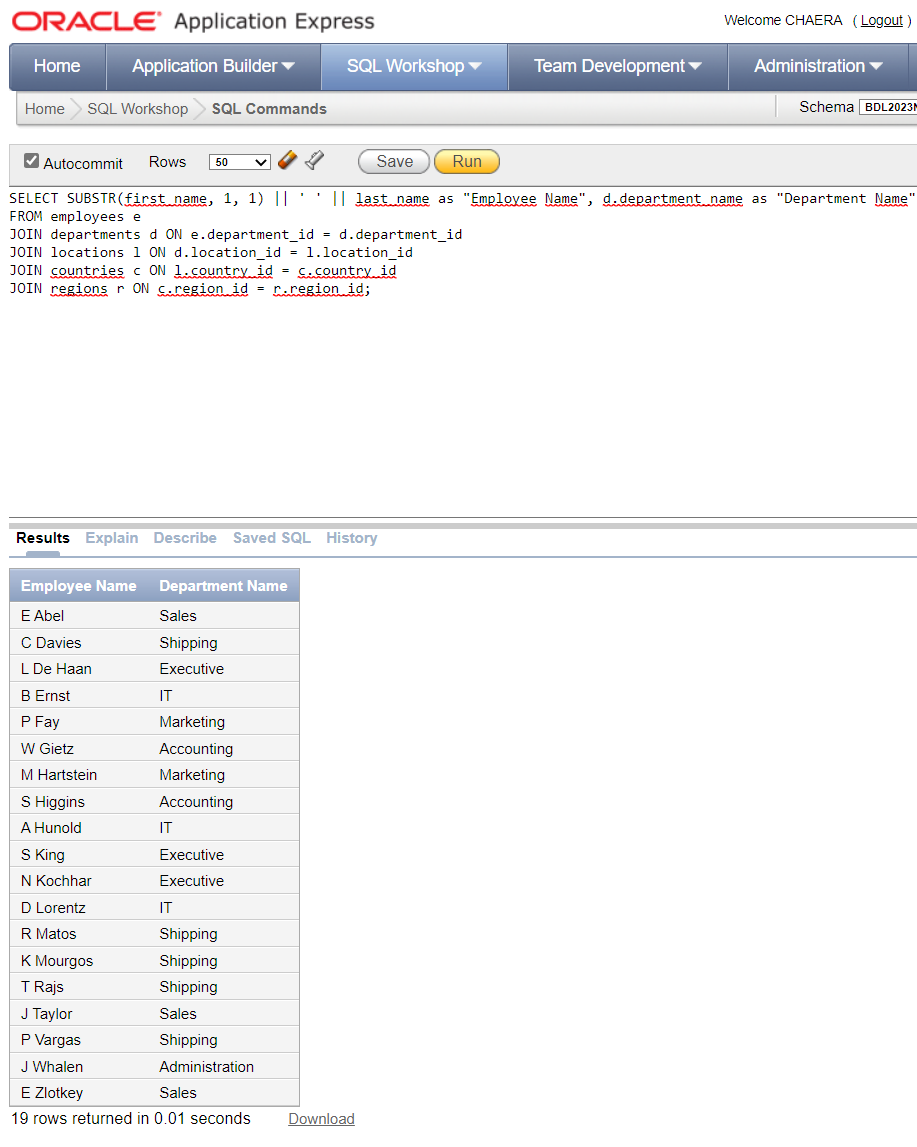
JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

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

JOIN regions r ON c.region\_id = r.region\_id;

Result :



1. Change the previous listing to join only on the department\_id column.

→ SELECT SUBSTR(e.first\_name, 1, 1) || ' ' || e.last\_name as "Employee Name",

d.department\_name as "Department Name"

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id;

Result :

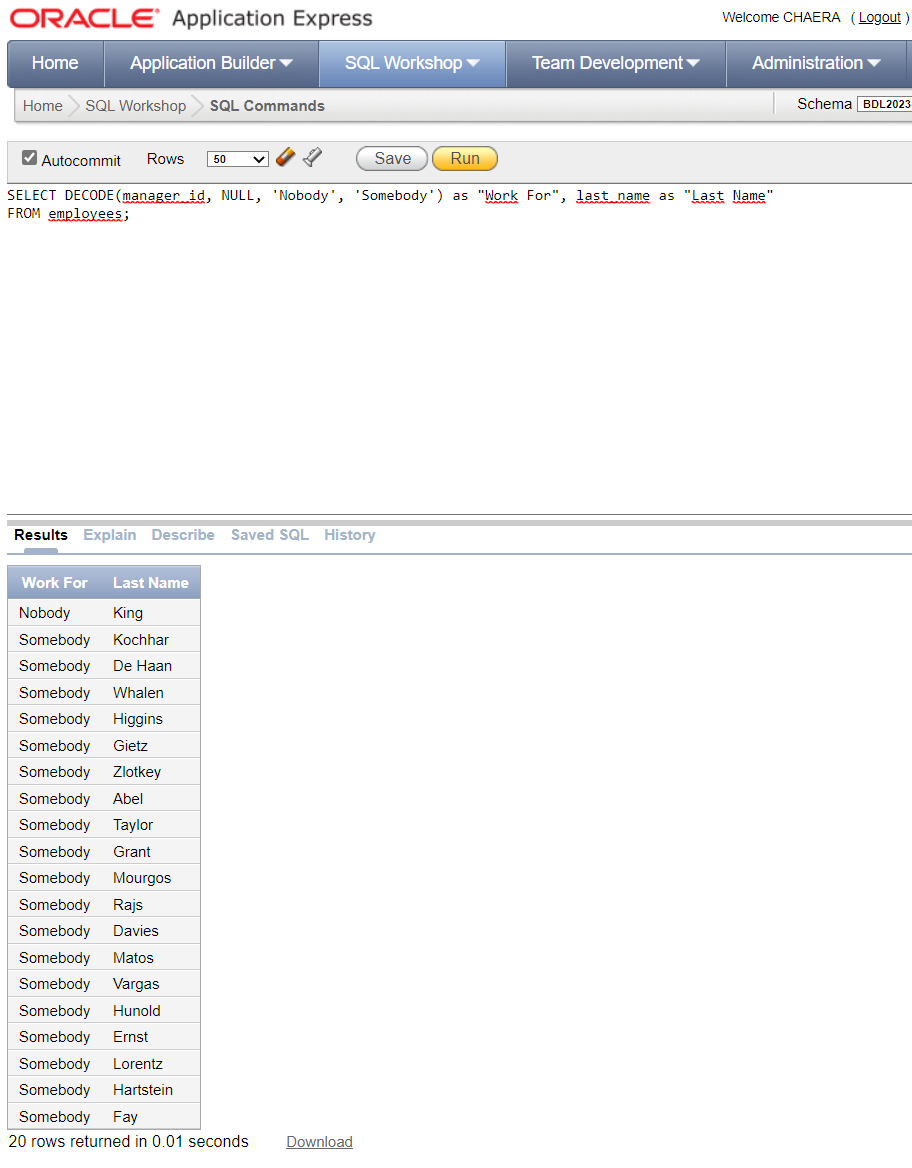


1. Create a list of every employee's last name, and the word nobody or somebody depending on whether or not the employee has a manager. Use the Oracle DECODE function to create the list.

→ SELECT DECODE(manager\_id, NULL, 'Nobody', 'Somebody') as "Work For", last\_name as "Last Name"

FROM employees;

Result :



1. Create a list of every employee's first initial and last name, salary, and a yes or no to show whether or not an employee makes a commission.

Fix this query to produce the result.

SELECT SUBSTR(first\_name,1 1)||' '|last\_name,

"Employee Name", salary "Salary",

DEC(commission\_pct NULL, 'No', 'Yes')'Commission'

FROM employees;

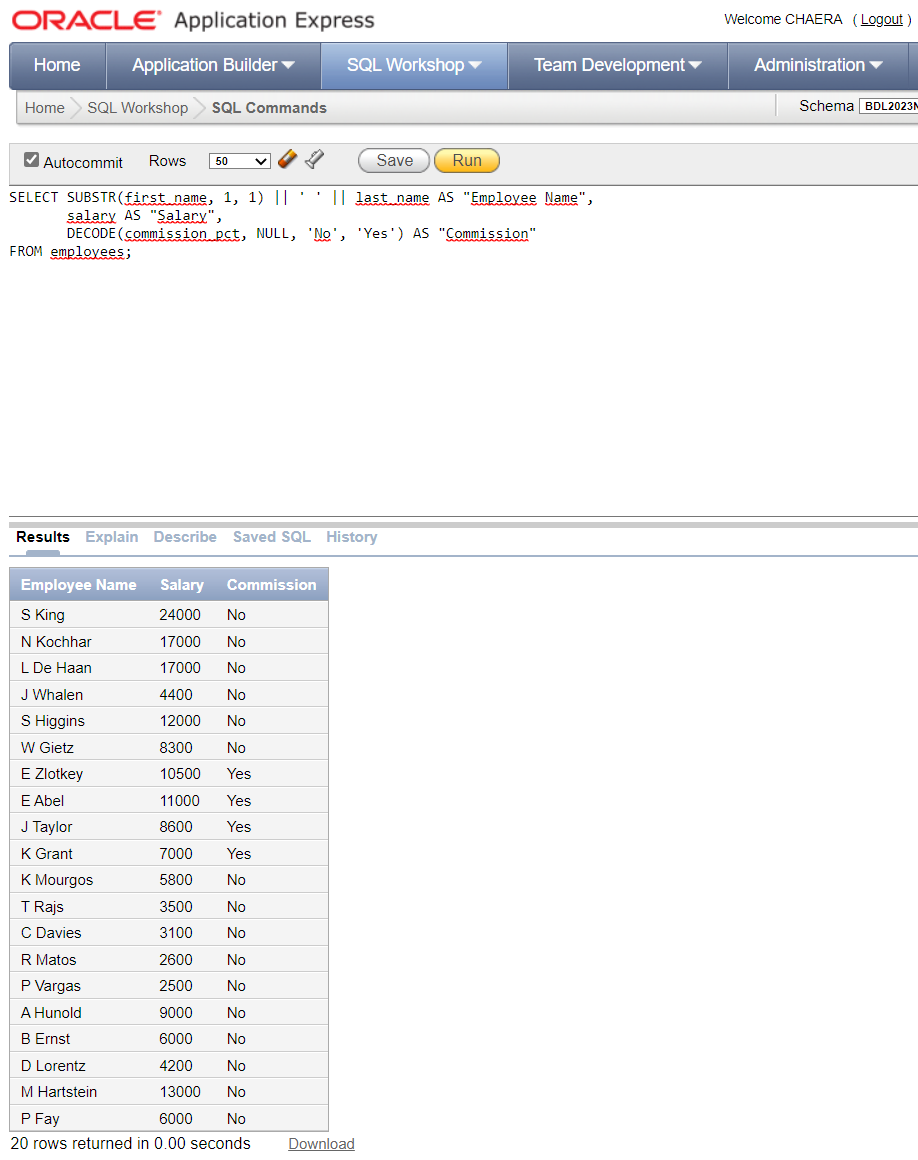
→ SELECT SUBSTR(first\_name, 1, 1) || ' ' || last\_name AS "Employee Name",

salary AS "Salary",

DECODE(commission\_pct, NULL, 'No', 'Yes') AS "Commission"

FROM employees;

Result :



1. Create a list of every employee's last name, department name, city, and state\_province.

Include departments without employees.

An outer join is required.

→ SELECT last\_name,

department\_name,

city,

state\_province

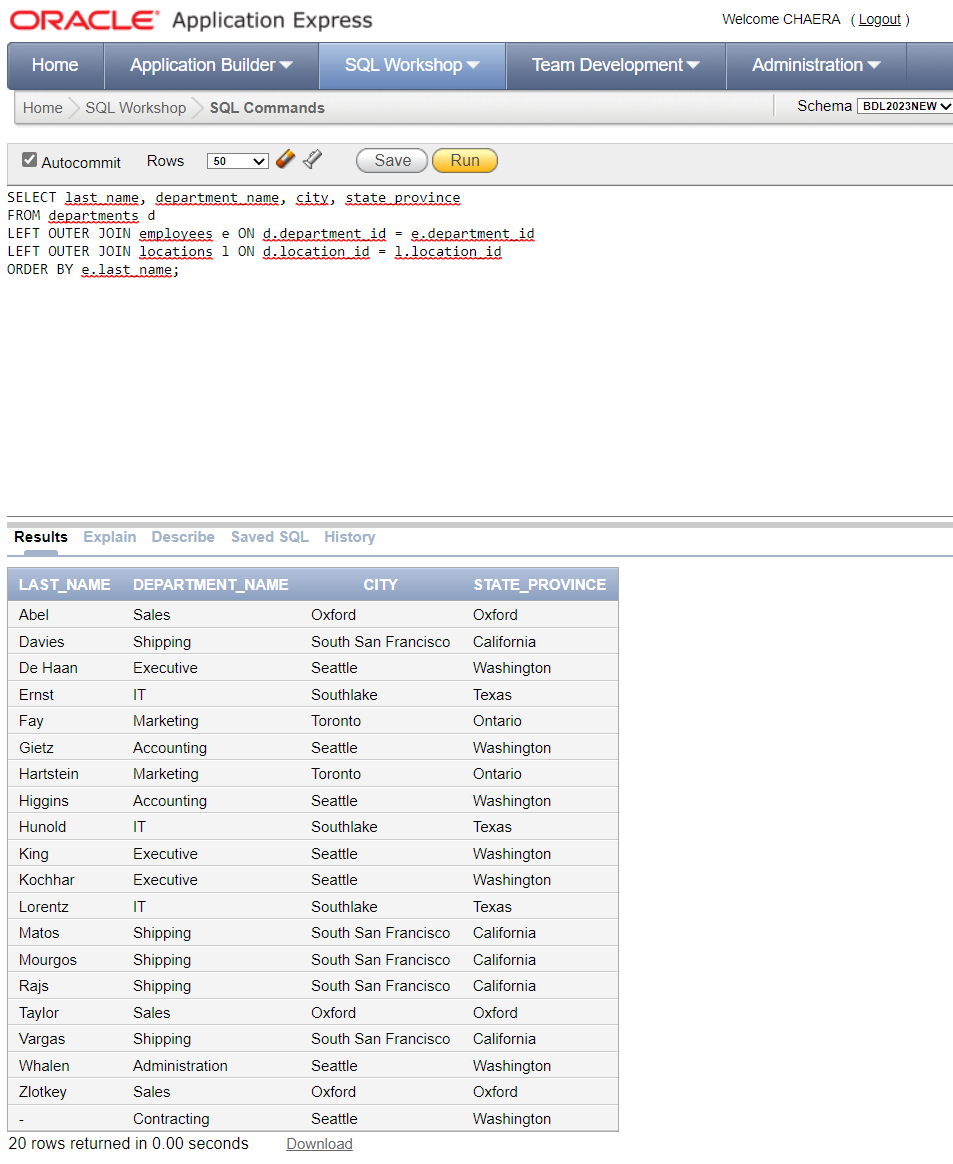
FROM departments d

LEFT OUTER JOIN employees e ON d.department\_id = e.department\_id

LEFT OUTER JOIN locations l ON d.location\_id = l.location\_id

ORDER BY e.last\_name;

Result :



1. Create a list of every employee's first and last names, and the first occurrence of: commission\_pct, manager\_id, or -1.

If an employee gets commission, display the commission\_pct column; if no commission, then display his manager\_id; if he has neither commission nor manager, then the number -1.

→ SELECT first\_name as "First Name",

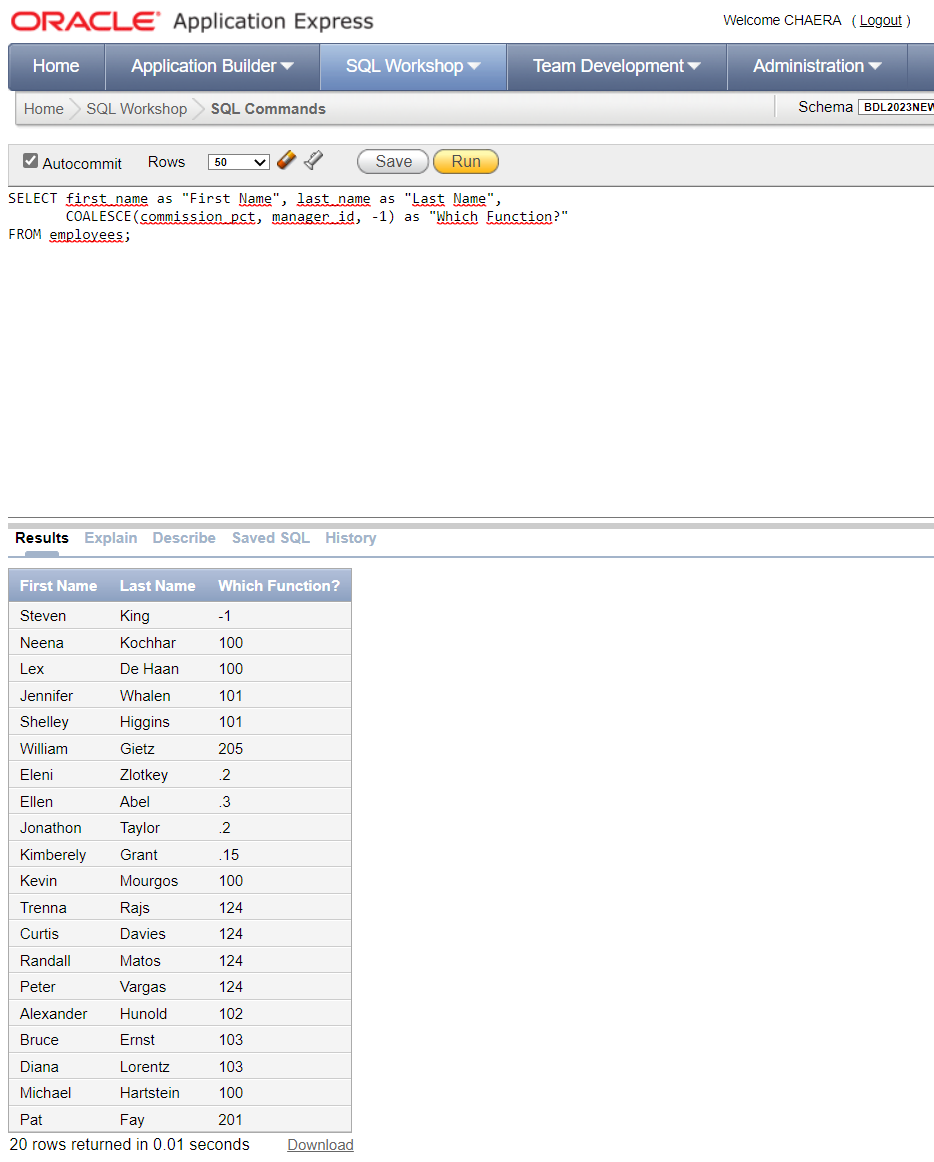
last\_name as "Last Name",

COALESCE(commission\_pct, manager\_id, -1)

as "Which Function?"

FROM employees;

Result :



1. Create a list of every employee's last name, salary, and job\_grade for all employees working in departments with a department\_id greater than 50.

→ SELECT e.last\_name, e.salary, jg.grade\_level

FROM employees e

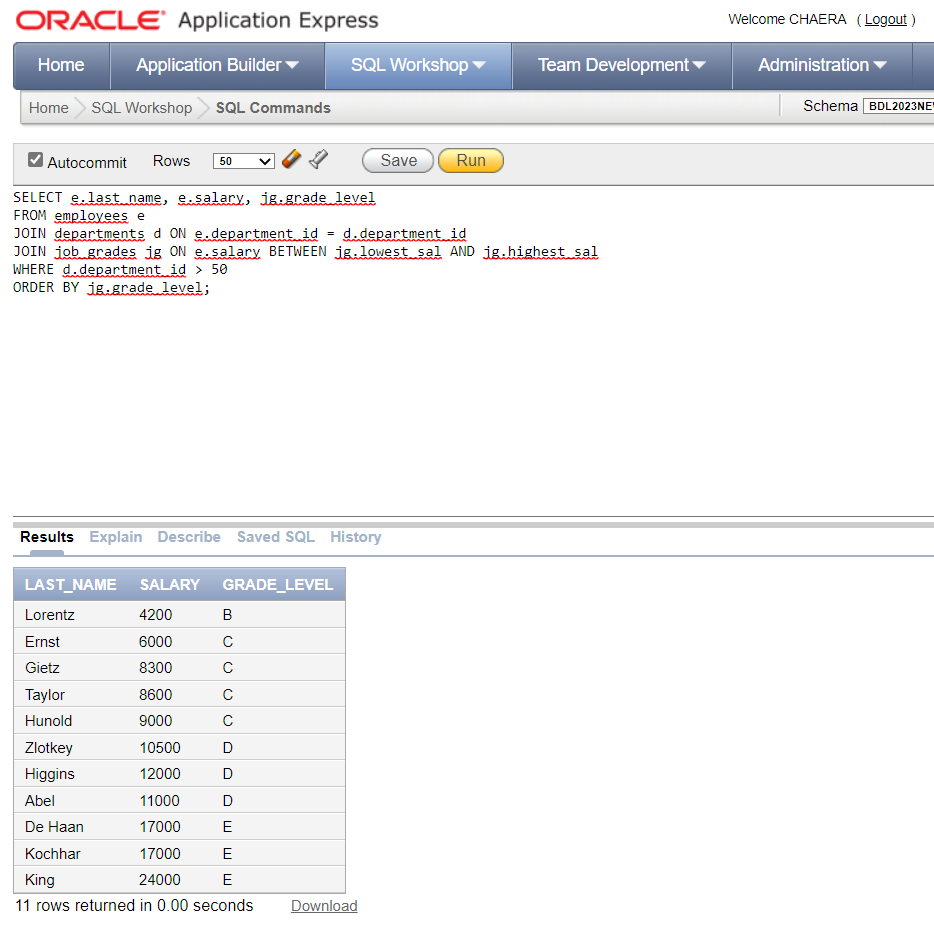
JOIN departments d ON e.department\_id = d.department\_id

JOIN job\_grades jg ON e.salary BETWEEN jg.lowest\_sal AND jg.highest\_sal

WHERE d.department\_id > 50

ORDER BY jg.grade\_level;

Result :



1. Produce a list of every employee's last name and department name. Include both employees without departments, and departments without employees.

→ SELECT last\_name, department\_name

FROM employees e

FULL OUTER JOIN departments d ON e.department\_id = d.department\_id

ORDER BY last\_name;

Result :



1. Create a treewalking list of every employee's last name, his manager’s last name, and his position in the company. The top level manager has position 1, this manager's subordinates position 2, their subordinates position 3, and so on.

Start the listing with employee number 100.

→ SELECT LEVEL as position, e.last\_name as employee\_name, m.last\_name as manager\_name

FROM employees e

LEFT JOIN employees m ON e.manager\_id = m.employee\_id

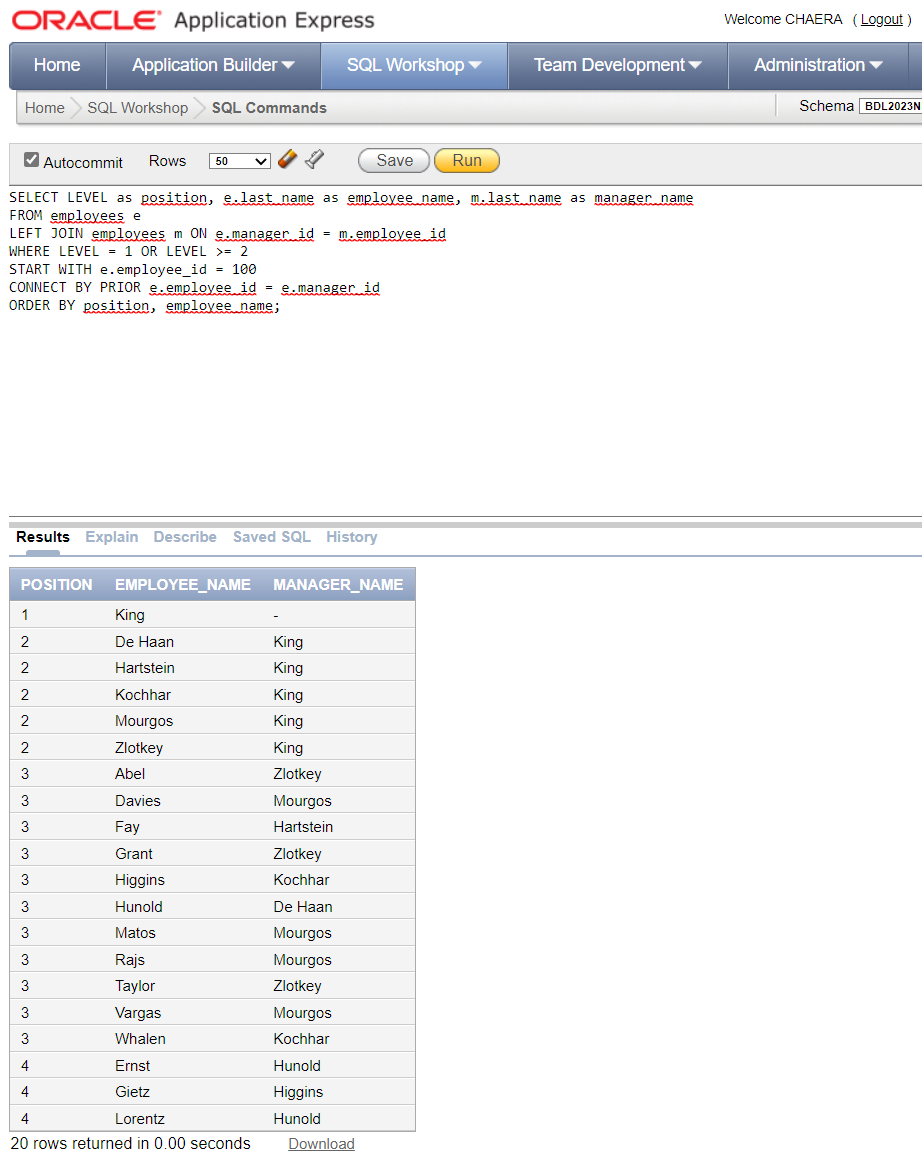
WHERE LEVEL = 1 OR LEVEL >= 2

START WITH e.employee\_id = 100

CONNECT BY PRIOR e.employee\_id = e.manager\_id

ORDER BY position, employee\_name;

Result :

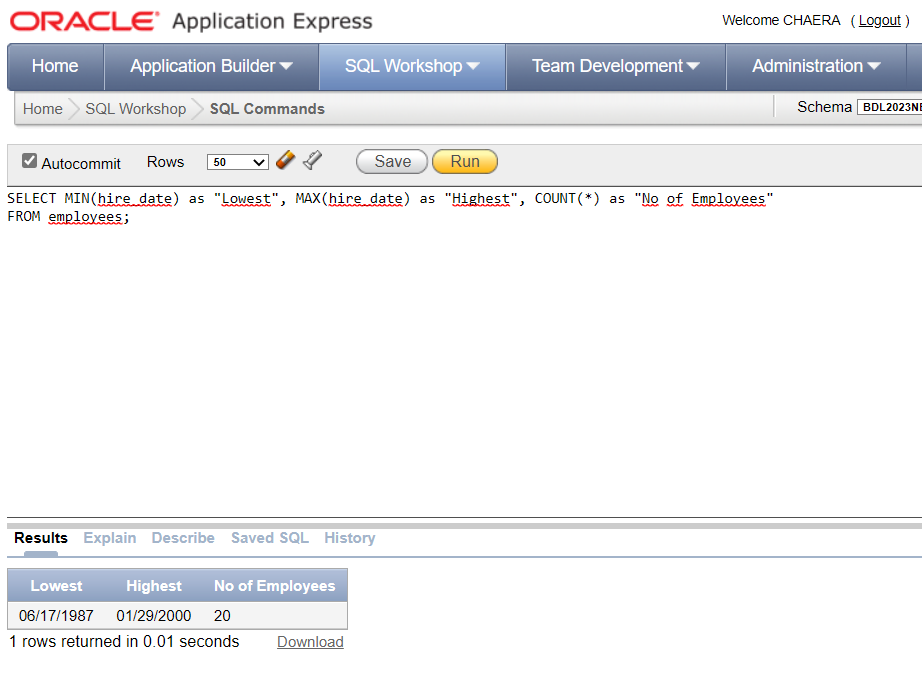


1. Produce a list of the earliest hire date, the latest hire date, and the number of employees from the employees table.

→ SELECT MIN(hire\_date) as "Lowest", MAX(hire\_date) as "Highest", COUNT(\*) as "No of Employees"

FROM employees;

Result :



1. Create a list of department names and the departmental costs (salaries added up). Include only departments whose salary costs are between 15000 and 31000, and sort the listing by the cost.

→ SELECT d.department\_name, SUM(e.salary) AS departmental\_cost

FROM employees e

INNER JOIN departments d

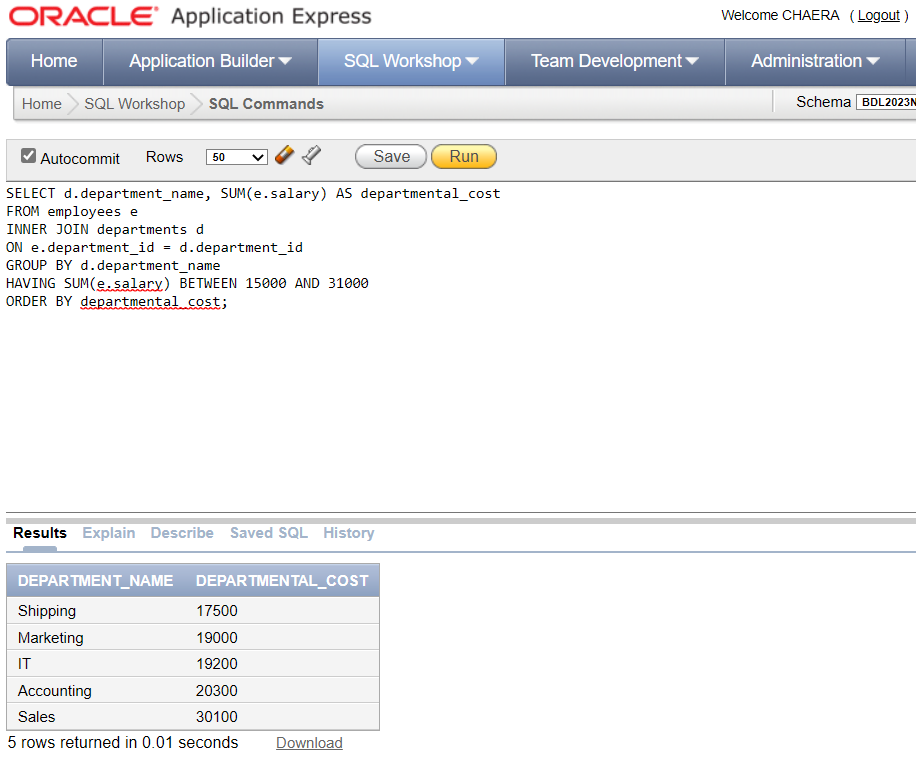
ON e.department\_id = d.department\_id

GROUP BY d.department\_name

HAVING SUM(e.salary) BETWEEN 15000 AND 31000

ORDER BY departmental\_cost;

Result :



1. Create a list of department names, the manager id, manager name (employee last name) of that department, and the average salary in each department.

→ SELECT d.department\_name, d.manager\_id, e.last\_name AS manager\_name, AVG(e.salary) AS avg\_salary

FROM employees e

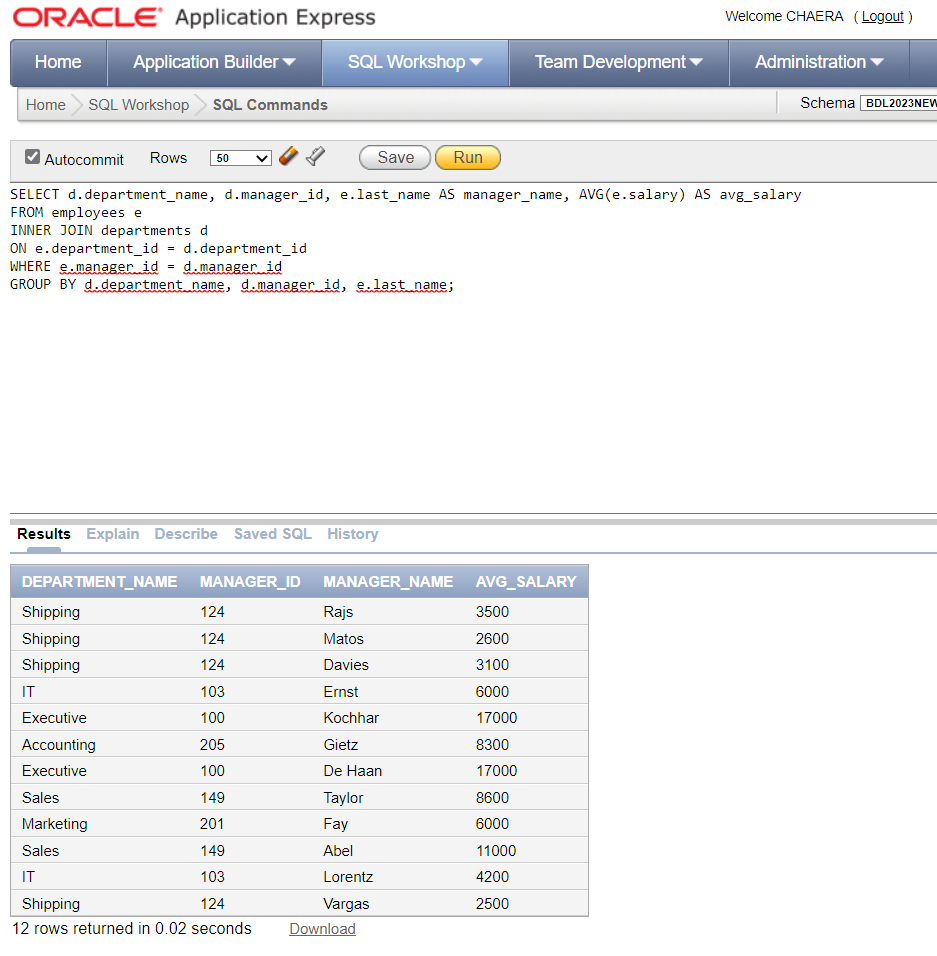
INNER JOIN departments d

ON e.department\_id = d.department\_id

WHERE e.manager\_id = d.manager\_id

GROUP BY d.department\_name, d.manager\_id, e.last\_name;

Result :



1. Show the highest average salary for the departments in the employees table. Round the result to the nearest whole number.

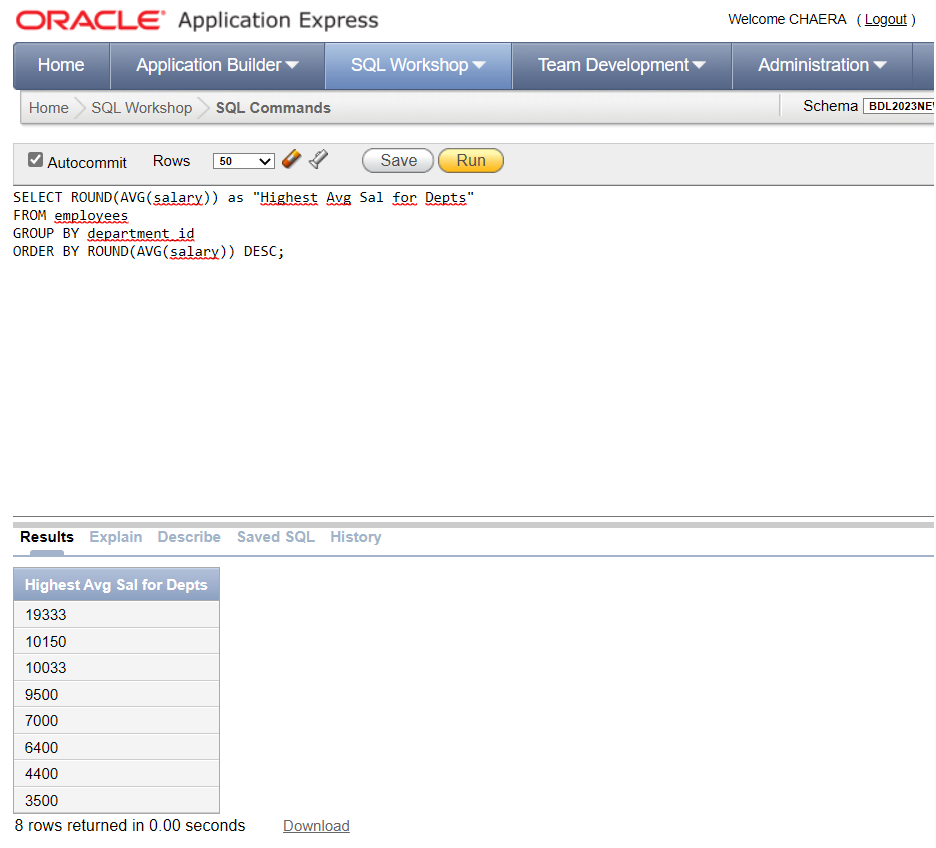
→ SELECT ROUND(AVG(salary)) as "Highest Avg Sal for Depts"

FROM employees

GROUP BY department\_id

ORDER BY ROUND(AVG(salary)) DESC;

Result :



1. Create a list of department names and their monthly costs (salaries added up).

→ SELECT d.department\_name as "Department Name",

SUM(e.salary) as "Monthly Cost"

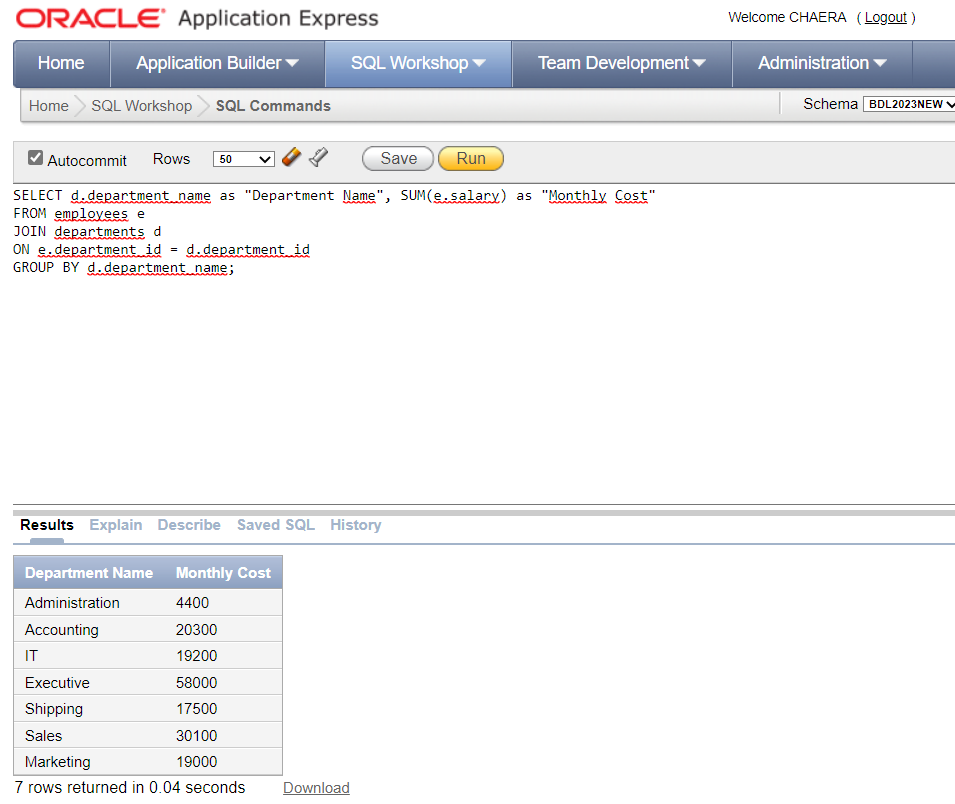
FROM employees e

JOIN departments d

ON e.department\_id = d.department\_id

GROUP BY d.department\_name;

Result :



1. Create a list of department names, and job\_ids. Calculate the monthly salary cost for each job\_id within a department, for each department, and for all departments added together.

→ SELECT department\_name as "Department Name",

job\_id as "Job Title",

SUM(salary) as "Monthly Cost"

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

GROUP BY GROUPING SETS((department\_name, job\_id), (department\_name))

ORDER BY department\_name, job\_id;

Result :



1. Create a list of department names, and job\_ids. Calculate the monthly salary cost for each job\_id within a department, for each department, for each group of job\_ids irrespective of the department, and for all departments added together (Hint: Cube).

→ SELECT department\_name as "Department Name",

job\_id as "Job Title",

SUM(salary) as "Monthly Cost"

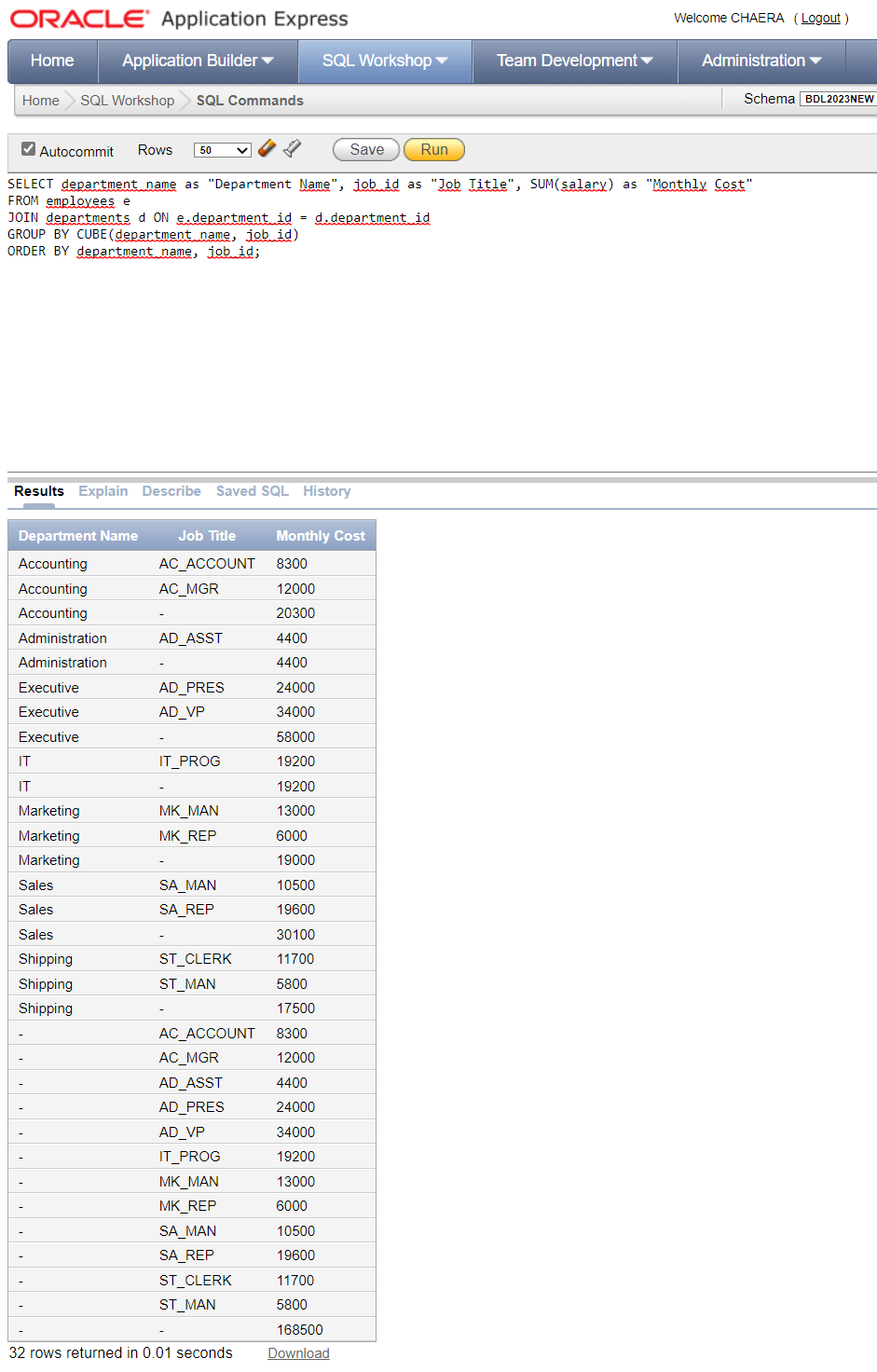
FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

GROUP BY CUBE(department\_name, job\_id)

ORDER BY department\_name, job\_id;

Result :



1. Expand the previous list to also show if the department\_id or job\_id was used to create the subtotals shown in the output (Hint: Cube, Grouping).

→ SELECT d.department\_name as "Department Name", e.job\_id as "Job Titlle", SUM(e.salary) as "Monthly Cost",

CASE WHEN GROUPING(d.department\_id) = 1 THEN 'Yes' ELSE 'No' END AS department\_id\_used,

CASE WHEN GROUPING(e.job\_id) = 1 THEN 'Yes' ELSE 'No' END AS job\_id\_used

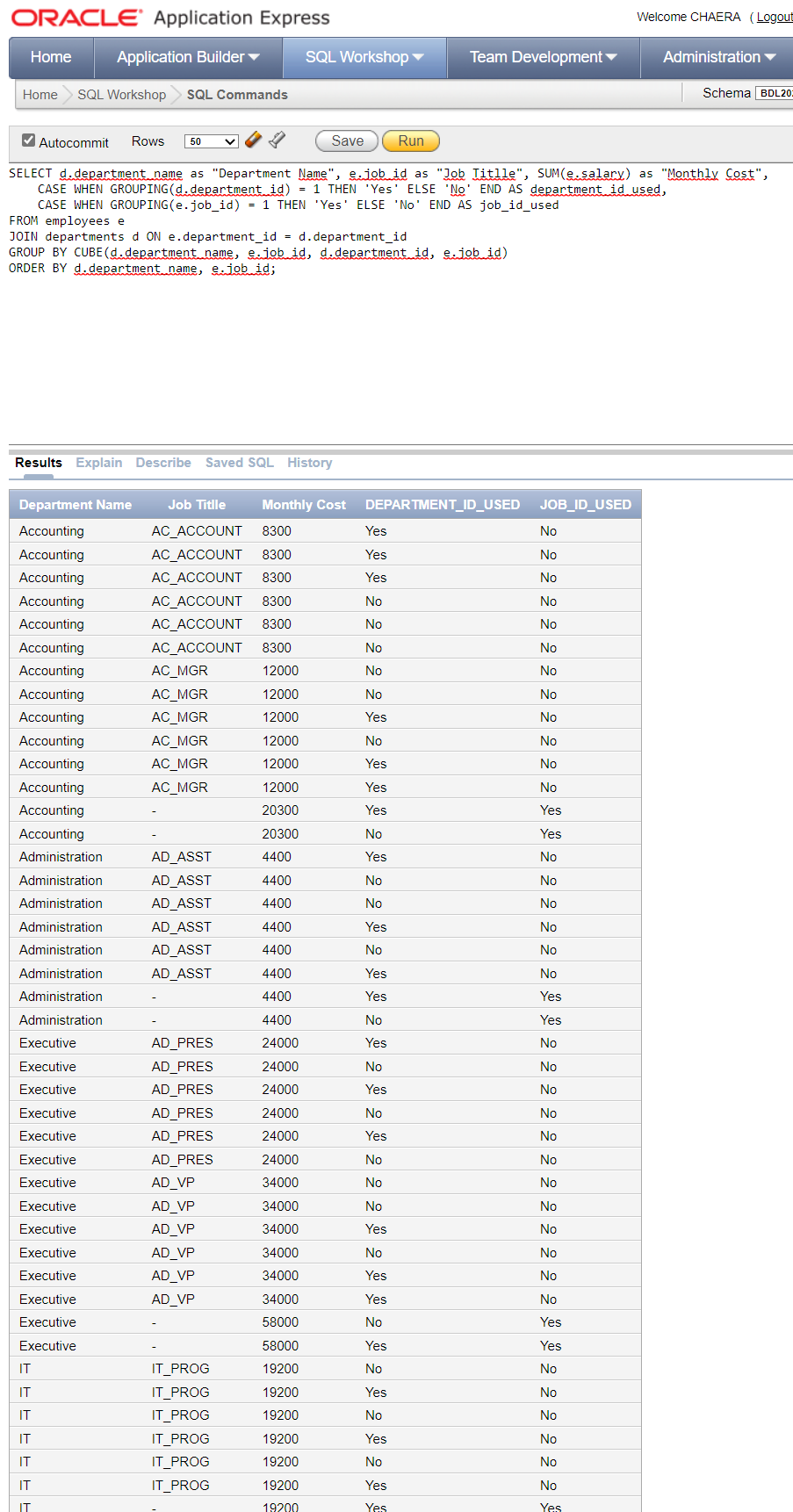
FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

GROUP BY CUBE(d.department\_name, e.job\_id, d.department\_id, e.job\_id)

ORDER BY d.department\_name, e.job\_id;

Result :



1. Create a list that includes the monthly salary costs for each job title within a department. In the same list, display the monthly salary cost per city. (Hint: Grouping Sets).

→ SELECT department\_name, job\_id, city, SUM(salary)

FROM employees e

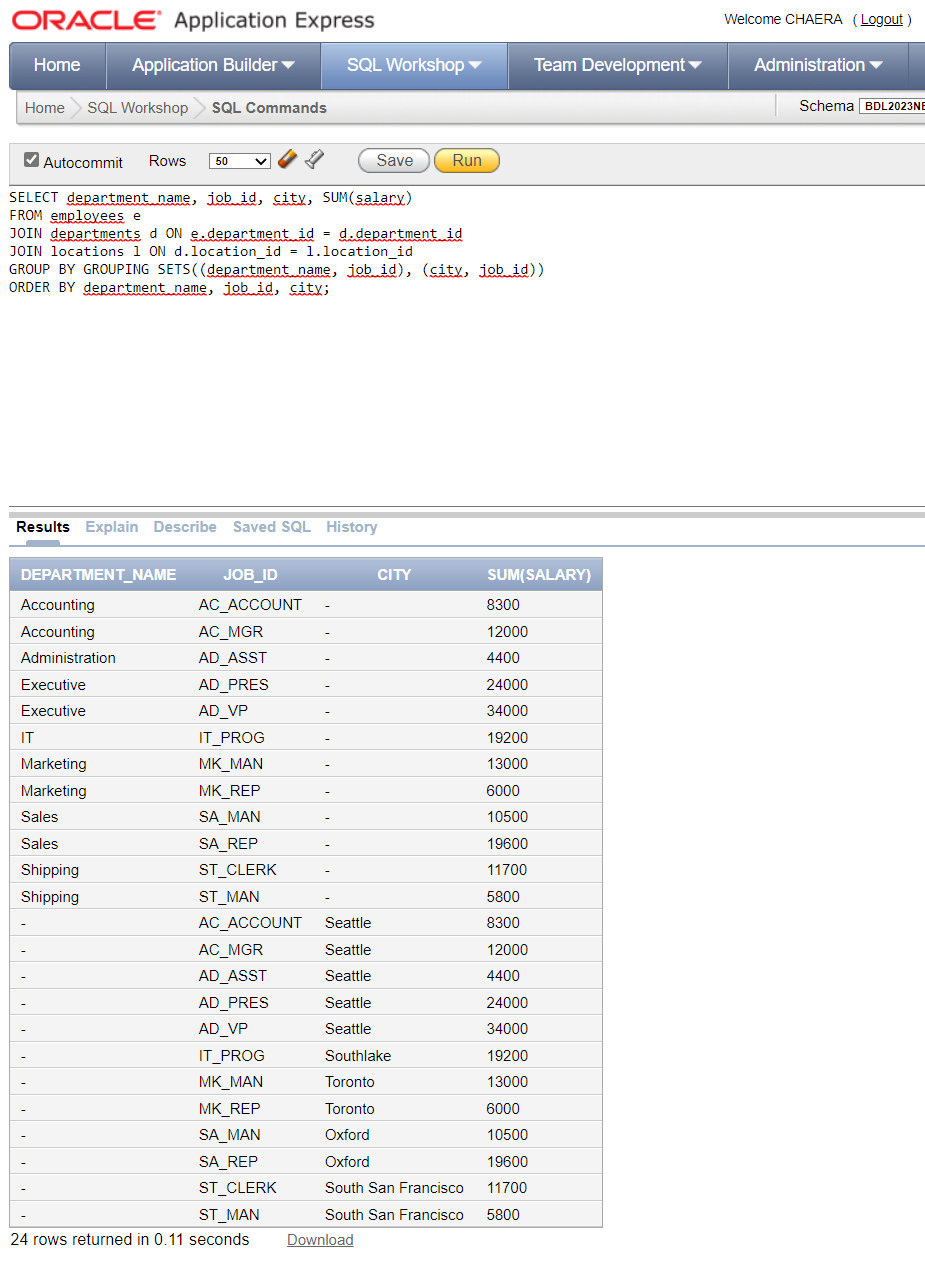
JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

GROUP BY GROUPING SETS((department\_name, job\_id), (city, job\_id))

ORDER BY department\_name, job\_id, city;

Result :



1. Create a list of employee names as shown and department ids. In the same report, list the department ids and department names. And finally, list the cities. The rows should not be joined, just listed in the same report. (Hint: Union).

→ SELECT employee\_id as "Employee ID", first\_name || ' ' || last\_name as "Employee Name", e.department\_id as "Department ID", NULL as "Department Name", NULL as "City"

FROM employees e

UNION

SELECT NULL AS employee\_id, NULL AS employee\_name, d.department\_id, d.department\_name, NULL AS city

FROM departments d

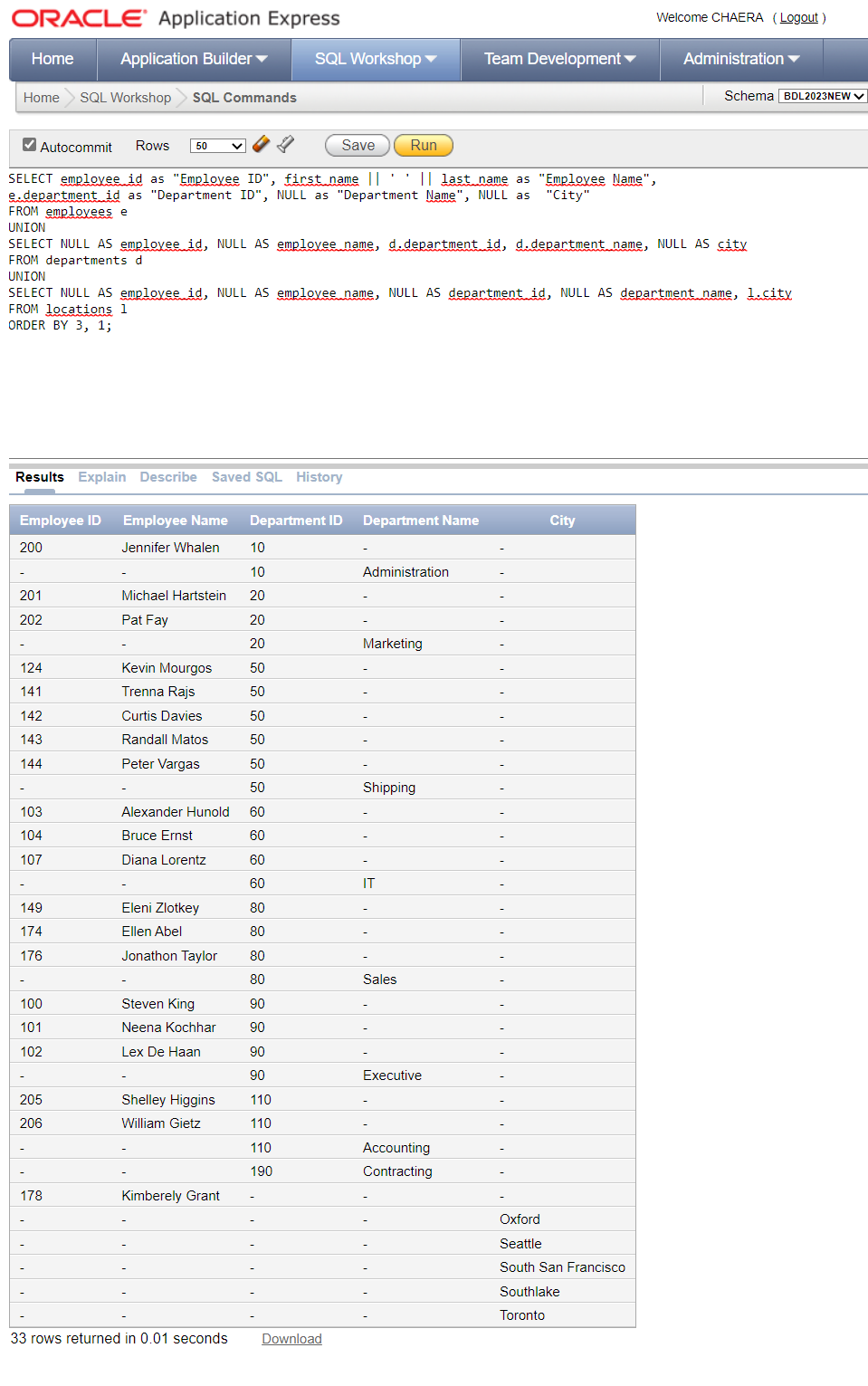
UNION

SELECT NULL AS employee\_id, NULL AS employee\_name, NULL AS department\_id, NULL AS department\_name, l.city

FROM locations l

ORDER BY 3, 1;

Result :



1. Create a list of each employee's first initial and last name, salary, and department name for each employee earning more than the average for his department.

→ SELECT SUBSTR(first\_name,1,1)|| '.' ||last\_name as "Employee", salary as "Salary", department\_name as "Department Name"

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

WHERE e.salary > (SELECT AVG(salary) FROM employees WHERE department\_id = e.department\_id)

ORDER BY d.department\_name, e.last\_name;

Result :

