Nama : Raden Sadiah Maharani

NIM : 2107126368

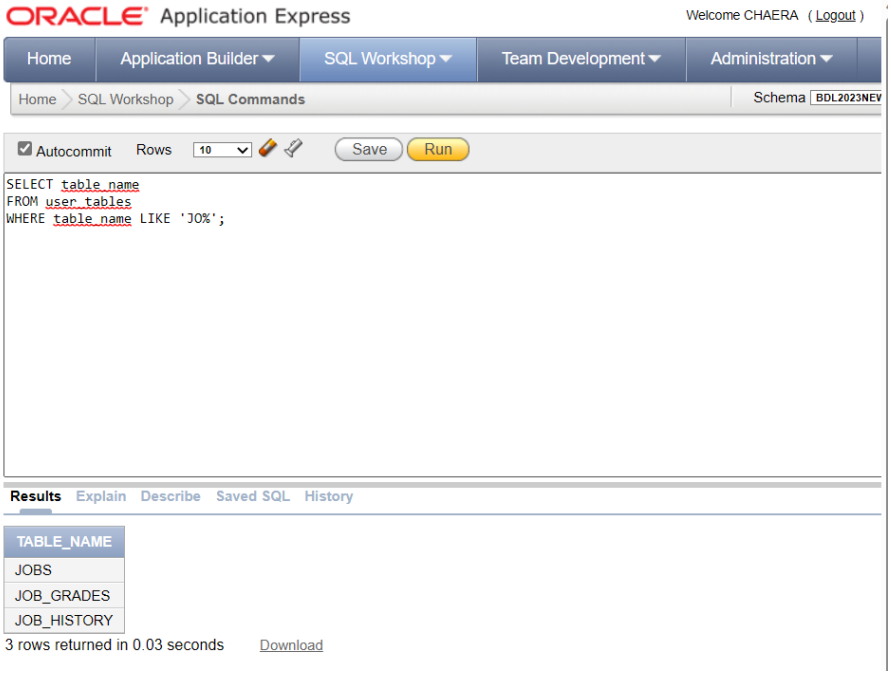
Basis Data Lanjut

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 :

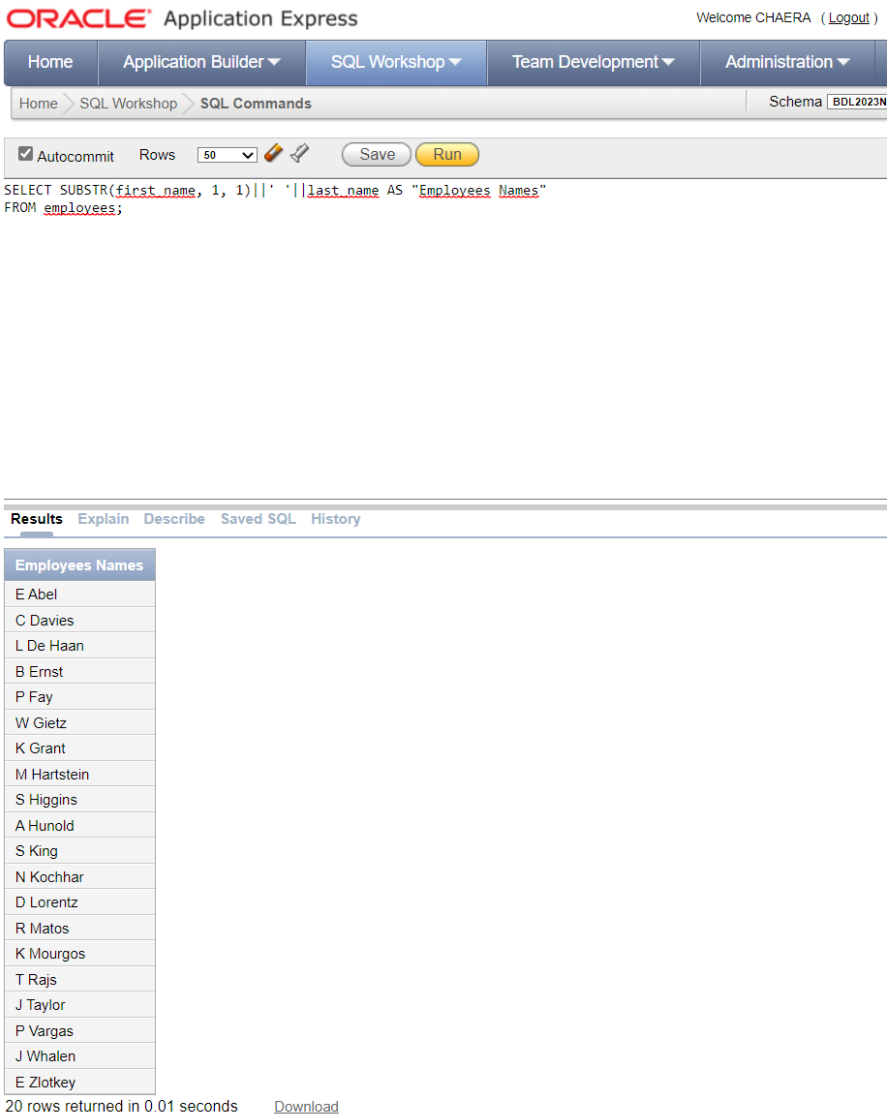


2. Create a list that includes the first initial of every employee's first name, a space, and the last name of the employee.

→ `SELECT SUBSTR(first_name, 1, 1)||' '||last_name AS "Employees Names"`

`FROM employees;`

Result :

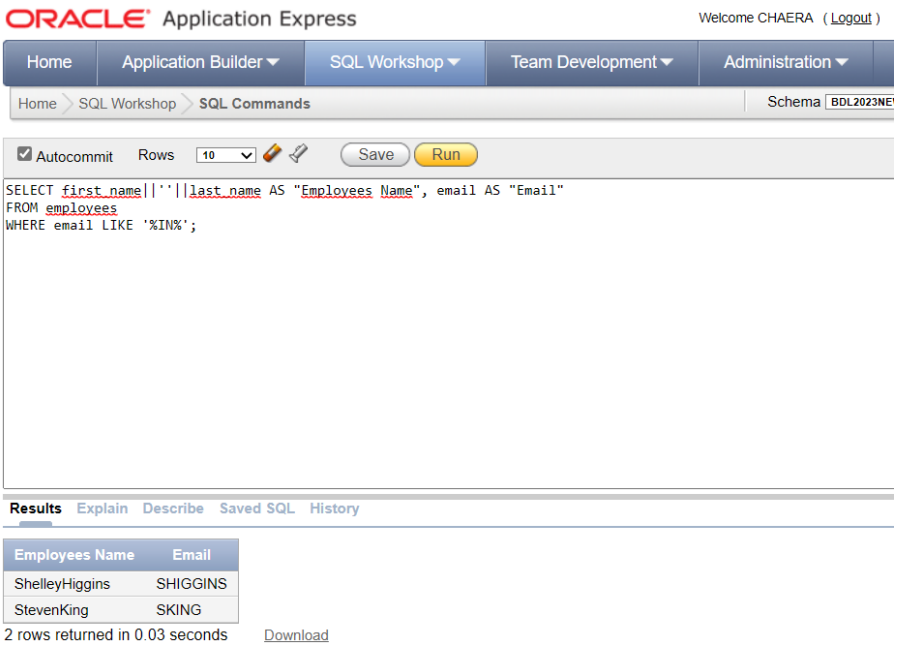


3. Create a list of every employee's first name concatenated to a space and the employee's last name, and the email of all employees where the email address contains the string 'IN'.

→ `SELECT first_name||''||last_name`

```
AS "Employees Name",
email AS "Email"
FROM employees
WHERE email LIKE '%IN%';
```

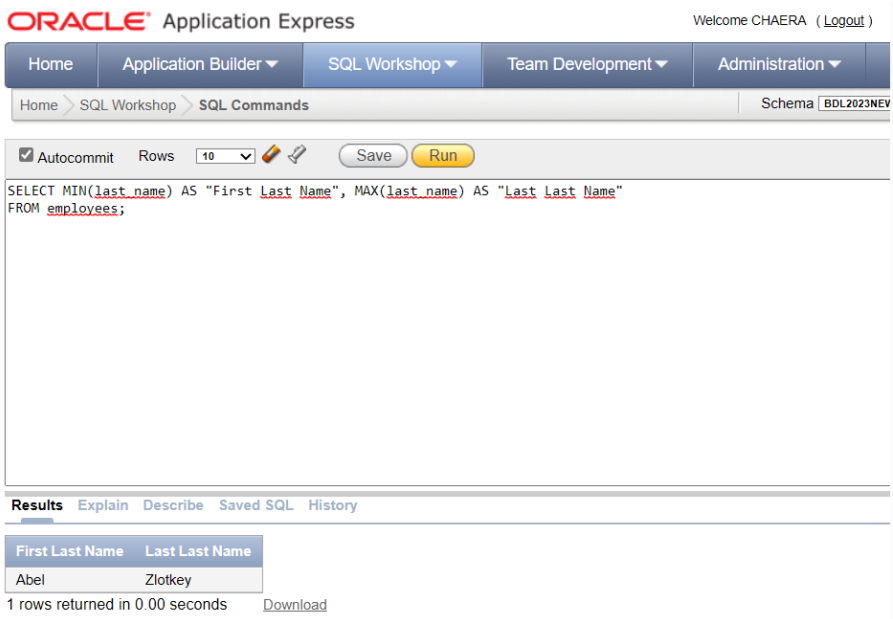
Result :



4. 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;
```

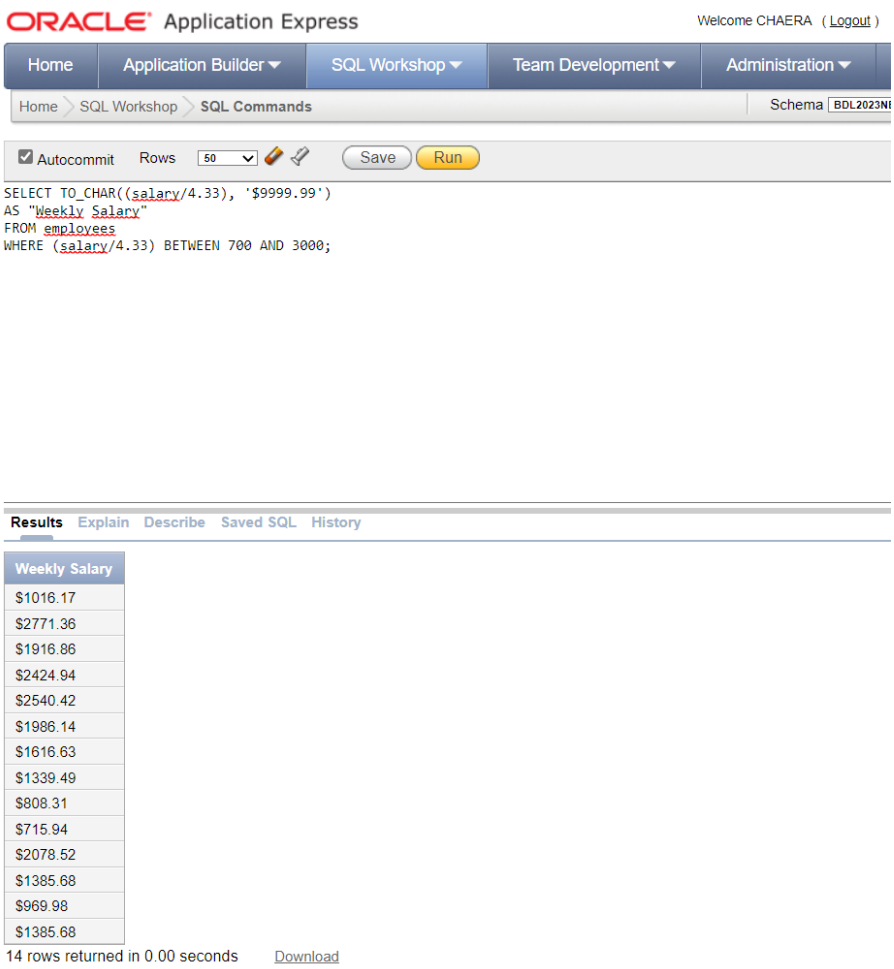
Result :



5. 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 :

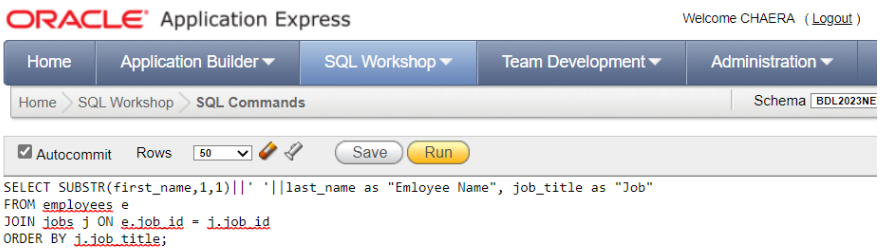


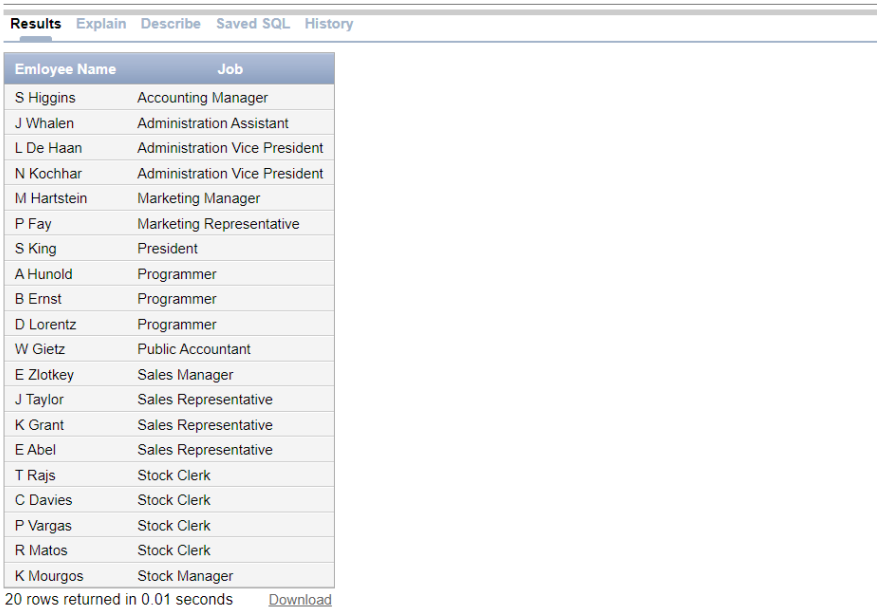
6. Create a list of every employee and his related job title sorted by job_title.

→ `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 :





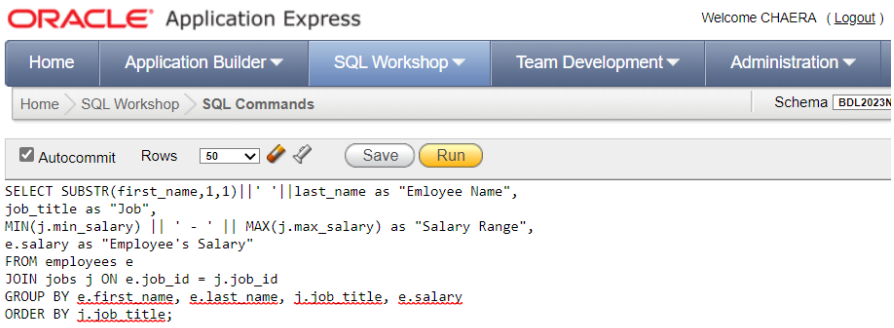
Emloyee Name	Job
S Higgins	Accounting Manager
J Whalen	Administration Assistant
L De Haan	Administration Vice President
N Kochhar	Administration Vice President
M Hartstein	Marketing Manager
P Fay	Marketing Representative
S King	President
A Hunold	Programmer
B Ernst	Programmer
D Lorentz	Programmer
W Gietz	Public Accountant
E Zlotkey	Sales Manager
J Taylor	Sales Representative
K Grant	Sales Representative
E Abel	Sales Representative
T Rajs	Stock Clerk
C Davies	Stock Clerk
P Vargas	Stock Clerk
R Matos	Stock Clerk
K Mourgos	Stock Manager

7. 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 :



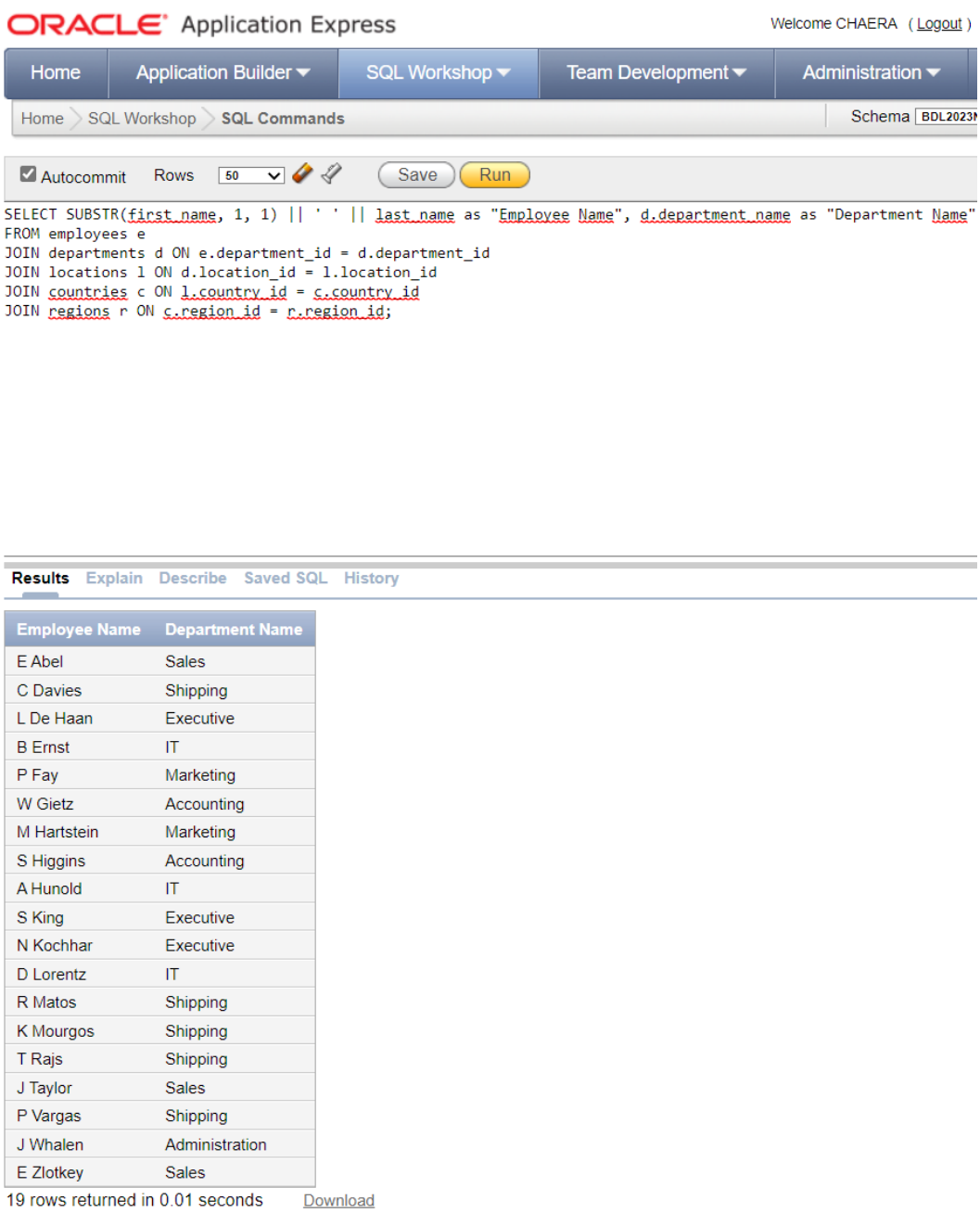
Emloyee Name	Job	Salary Range	Employee's Salary
S Higgins	Accounting Manager	8200 - 16000	12000
J Whalen	Administration Assistant	3000 - 6000	4400
L De Haan	Administration Vice President	15000 - 30000	17000
N Kochhar	Administration Vice President	15000 - 30000	17000
M Hartstein	Marketing Manager	9000 - 15000	13000
P Fay	Marketing Representative	4000 - 9000	6000
S King	President	20000 - 40000	24000
B Ernst	Programmer	4000 - 10000	6000
A Hunold	Programmer	4000 - 10000	9000
D Lorentz	Programmer	4000 - 10000	4200
W Gietz	Public Accountant	4200 - 9000	8300
E Zlotkey	Sales Manager	10000 - 20000	10500
E Abel	Sales Representative	6000 - 12000	11000
K Grant	Sales Representative	6000 - 12000	7000
J Taylor	Sales Representative	6000 - 12000	8600
C Davies	Stock Clerk	2000 - 5000	3100
R Matos	Stock Clerk	2000 - 5000	2600
T Rajs	Stock Clerk	2000 - 5000	3500
P Vargas	Stock Clerk	2000 - 5000	2500
K Mourgos	Stock Manager	5500 - 8500	5800

8. 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 :



9. Change the previous listing to join only on the department_id column.

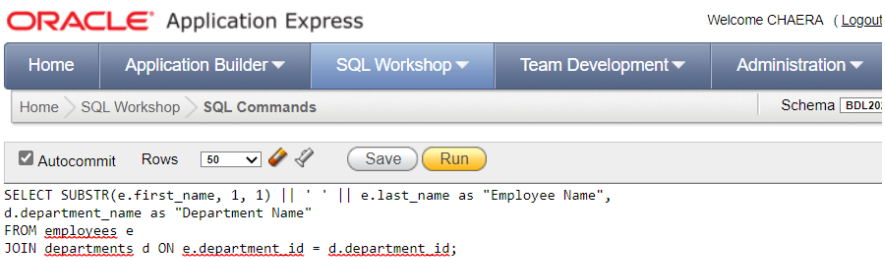
→ `SELECT SUBSTR(e.first_name, 1, 1) || ' ' || e.last_name as "Employee Name",`

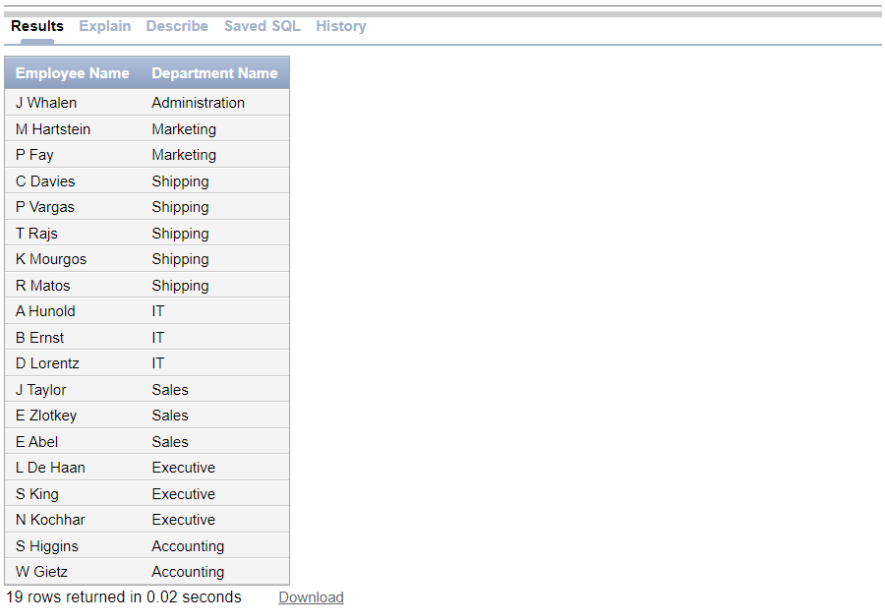
`d.department_name as "Department Name"`

`FROM employees e`

```
JOIN departments d ON e.department_id =
d.department_id;
```

Result :





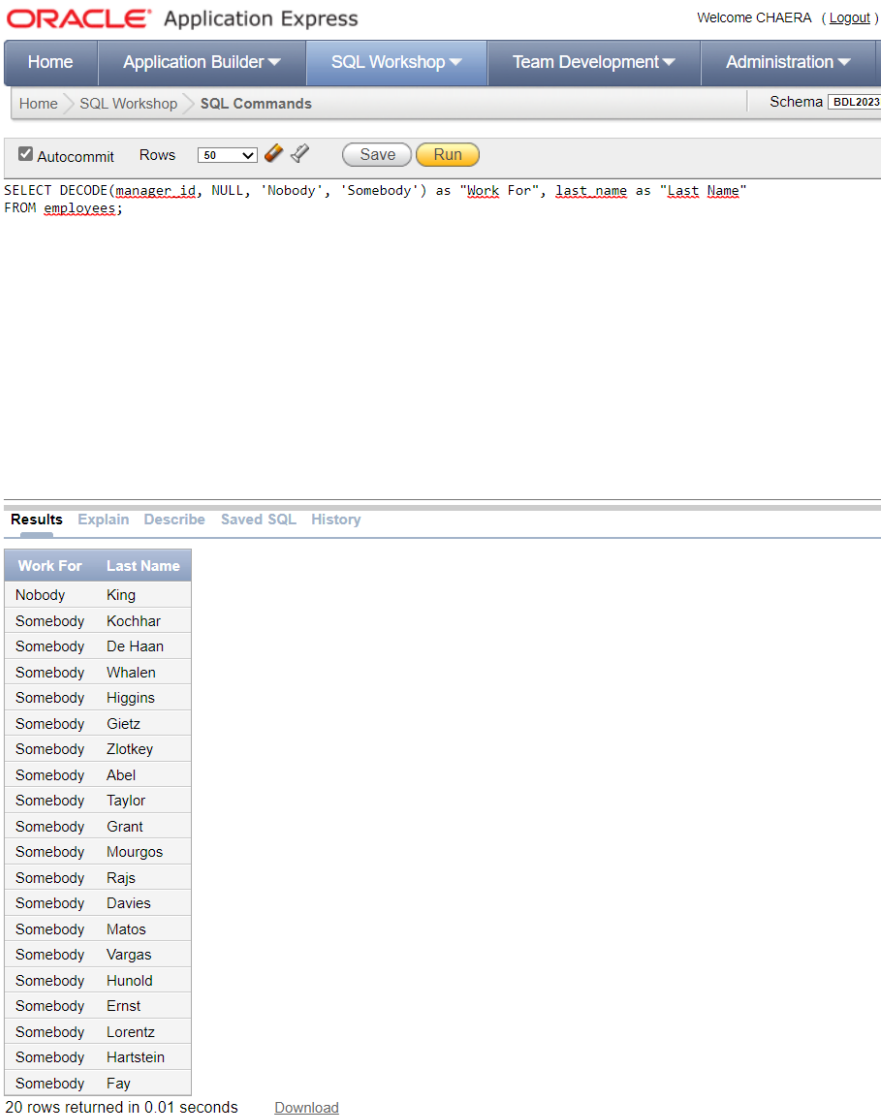
Employee Name	Department Name
J Whalen	Administration
M Hartstein	Marketing
P Fay	Marketing
C Davies	Shipping
P Vargas	Shipping
T Rajs	Shipping
K Mourgos	Shipping
R Matos	Shipping
A Hunold	IT
B Ernst	IT
D Lorentz	IT
J Taylor	Sales
E Zlotkey	Sales
E Abel	Sales
L De Haan	Executive
S King	Executive
N Kochhar	Executive
S Higgins	Accounting
W Gietz	Accounting

10. 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 :



Work For	Last Name
Nobody	King
Somebody	Kochhar
Somebody	De Haan
Somebody	Whalen
Somebody	Higgins
Somebody	Gietz
Somebody	Zlotkey
Somebody	Abel
Somebody	Taylor
Somebody	Grant
Somebody	Mourgos
Somebody	Rajs
Somebody	Davies
Somebody	Matos
Somebody	Vargas
Somebody	Hunold
Somebody	Ernst
Somebody	Lorentz
Somebody	Hartstein
Somebody	Fay

11. 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;
```

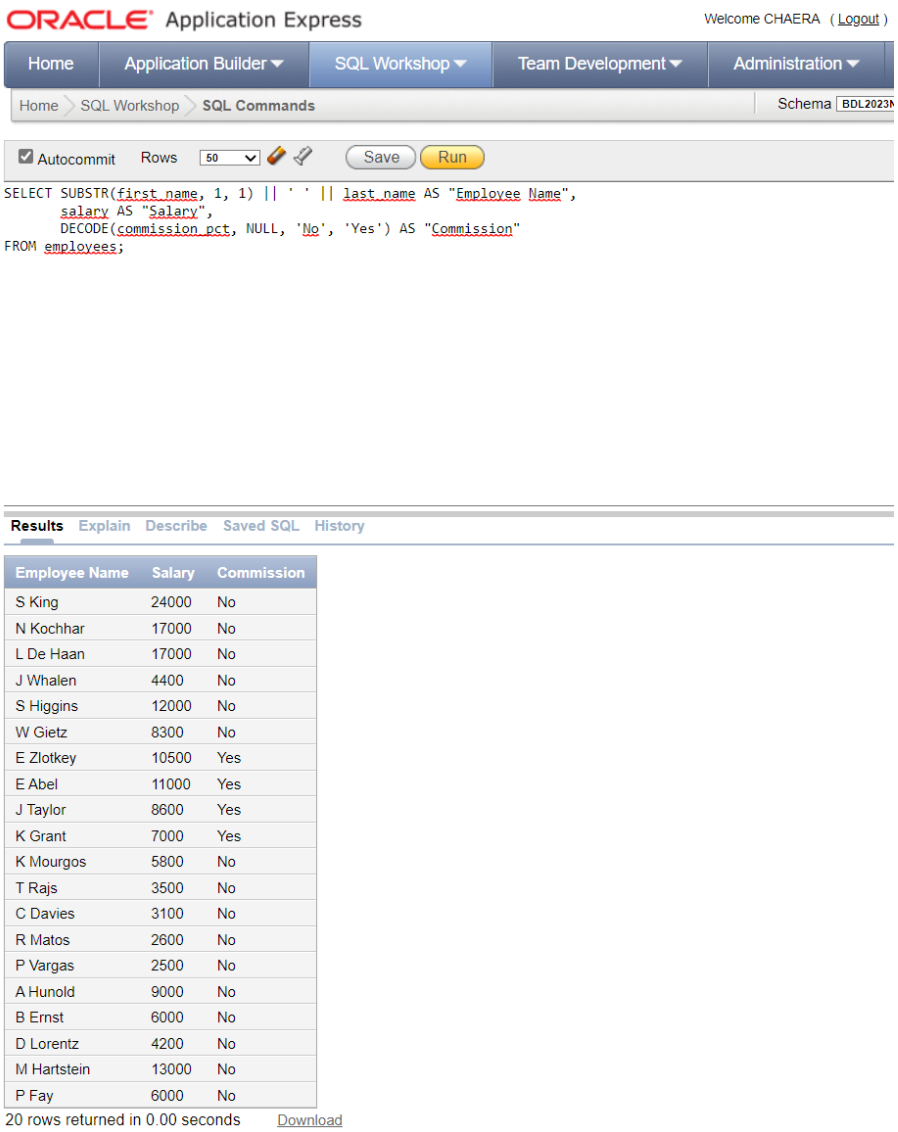
→ SELECT SUBSTR(first_name, 1, 1) || ' ' || last_name AS "Employee Name",

salary AS "Salary",

DECODE(commission_pct, NULL, 'No', 'Yes') AS "Commission"

FROM employees;

Result :



Employee Name	Salary	Commission
S King	24000	No
N Kochhar	17000	No
L De Haan	17000	No
J Whalen	4400	No
S Higgins	12000	No
W Gietz	8300	No
E Zlotkey	10500	Yes
E Abel	11000	Yes
J Taylor	8600	Yes
K Grant	7000	Yes
K Mourgos	5800	No
T Rajs	3500	No
C Davies	3100	No
R Matos	2600	No
P Vargas	2500	No
A Hunold	9000	No
B Ernst	6000	No
D Lorentz	4200	No
M Hartstein	13000	No
P Fay	6000	No

12. Create a list of every employee's last name, department name, city, and state_province.

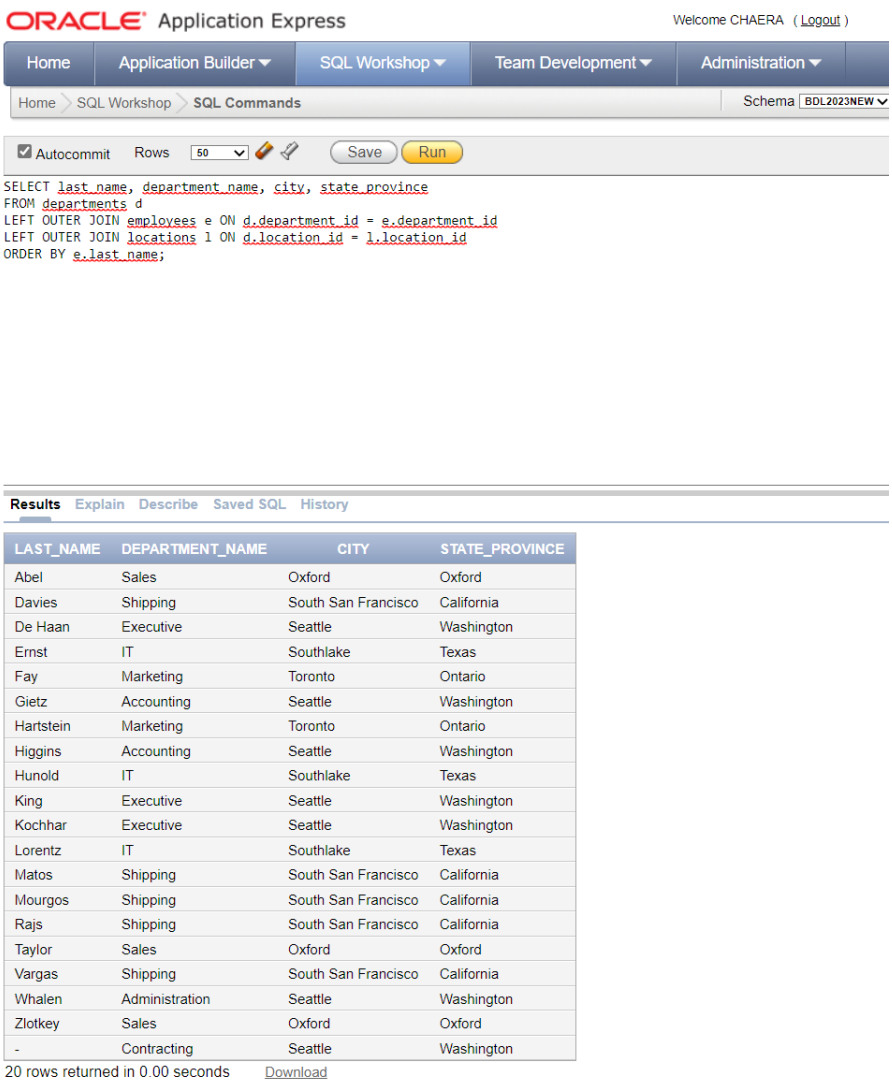
    Include departments without employees.

    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 :



LAST_NAME	DEPARTMENT_NAME	CITY	STATE_PROVINCE
Abel	Sales	Oxford	Oxford
Davies	Shipping	South San Francisco	California
De Haan	Executive	Seattle	Washington
Ernst	IT	Southlake	Texas
Fay	Marketing	Toronto	Ontario
Gietz	Accounting	Seattle	Washington
Hartstein	Marketing	Toronto	Ontario
Higgins	Accounting	Seattle	Washington
Hunold	IT	Southlake	Texas
King	Executive	Seattle	Washington
Kochhar	Executive	Seattle	Washington
Lorentz	IT	Southlake	Texas
Matos	Shipping	South San Francisco	California
Mourgos	Shipping	South San Francisco	California
Rajs	Shipping	South San Francisco	California
Taylor	Sales	Oxford	Oxford
Vargas	Shipping	South San Francisco	California
Whalen	Administration	Seattle	Washington
Zlotkey	Sales	Oxford	Oxford
-	Contracting	Seattle	Washington

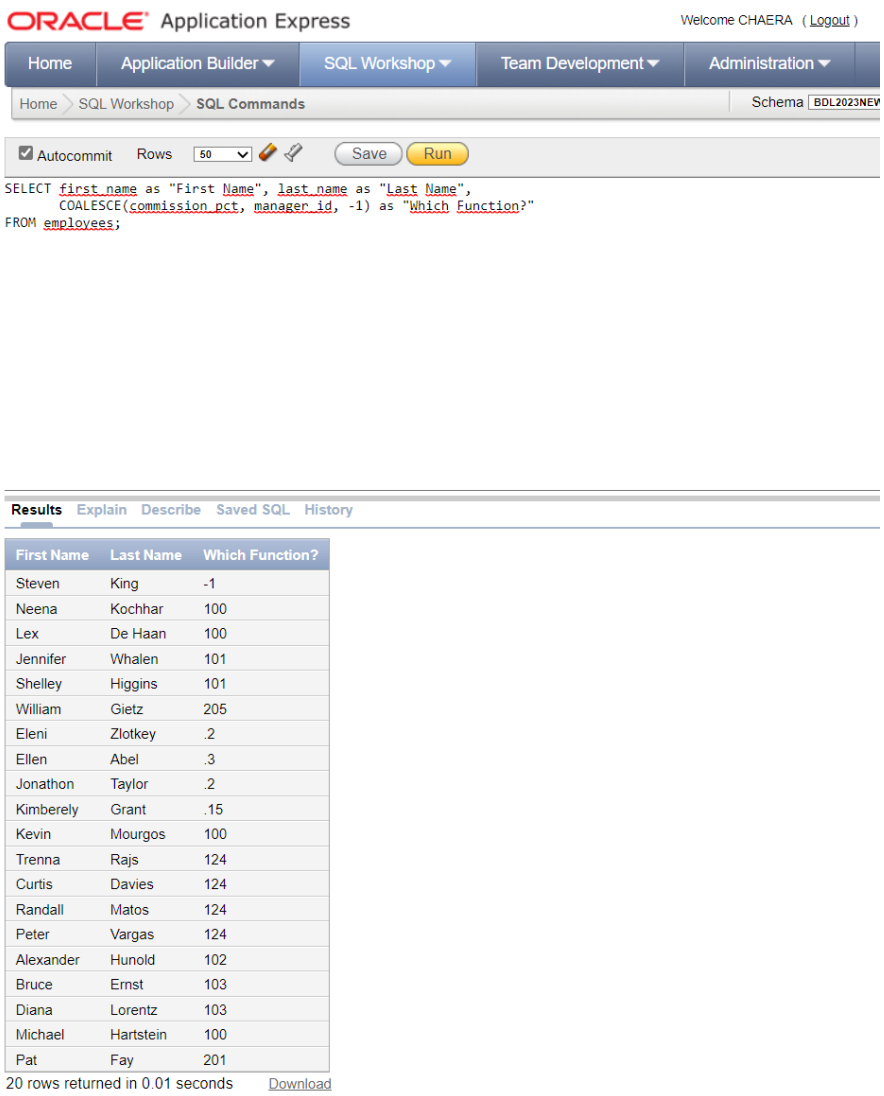
13. Create a list of every employee's first and last names, and the first occurrence of: commission_pct, manager_id, or -1.

    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 :



First Name	Last Name	Which Function?
Steven	King	-1
Neena	Kochhar	100
Lex	De Haan	100
Jennifer	Whalen	101
Shelley	Higgins	101
William	Gietz	205
Eleni	Zlotkey	.2
Ellen	Abel	.3
Jonathon	Taylor	.2
Kimberely	Grant	.15
Kevin	Mourgos	100
Trenna	Rajs	124
Curtis	Davies	124
Randall	Matos	124
Peter	Vargas	124
Alexander	Hunold	102
Bruce	Ernst	103
Diana	Lorentz	103
Michael	Hartstein	100
Pat	Fay	201

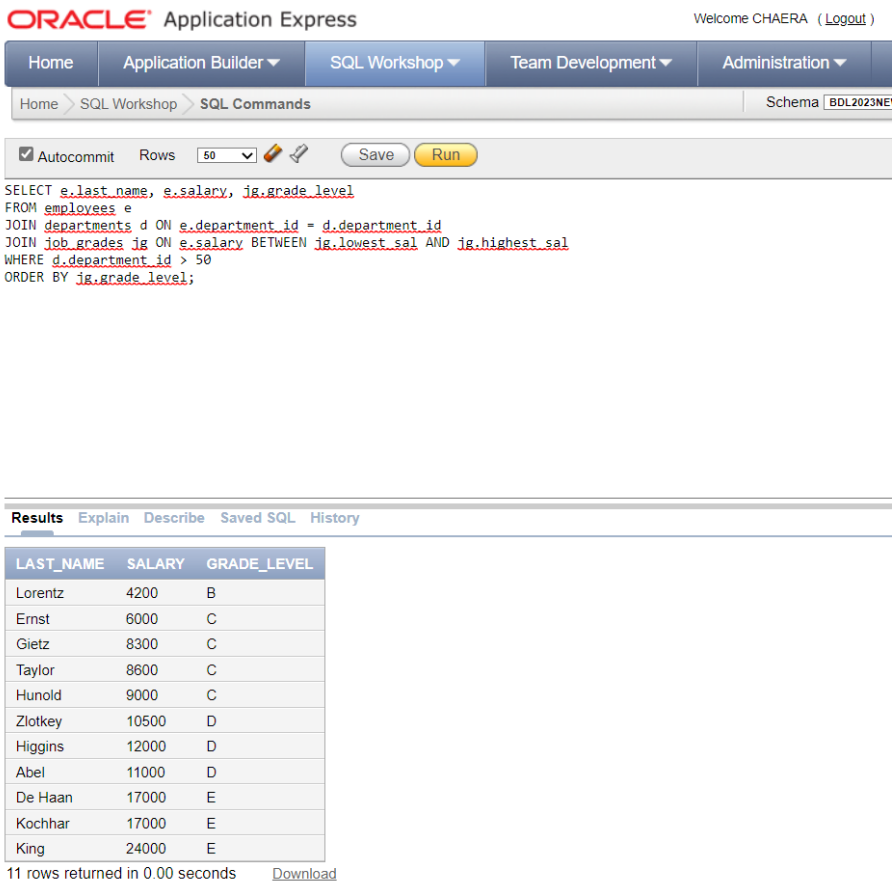
14. Create a list of every employee's last name, salary, and job_grade for all employees working in departments with a department_id greater than 50.

→

```
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 :

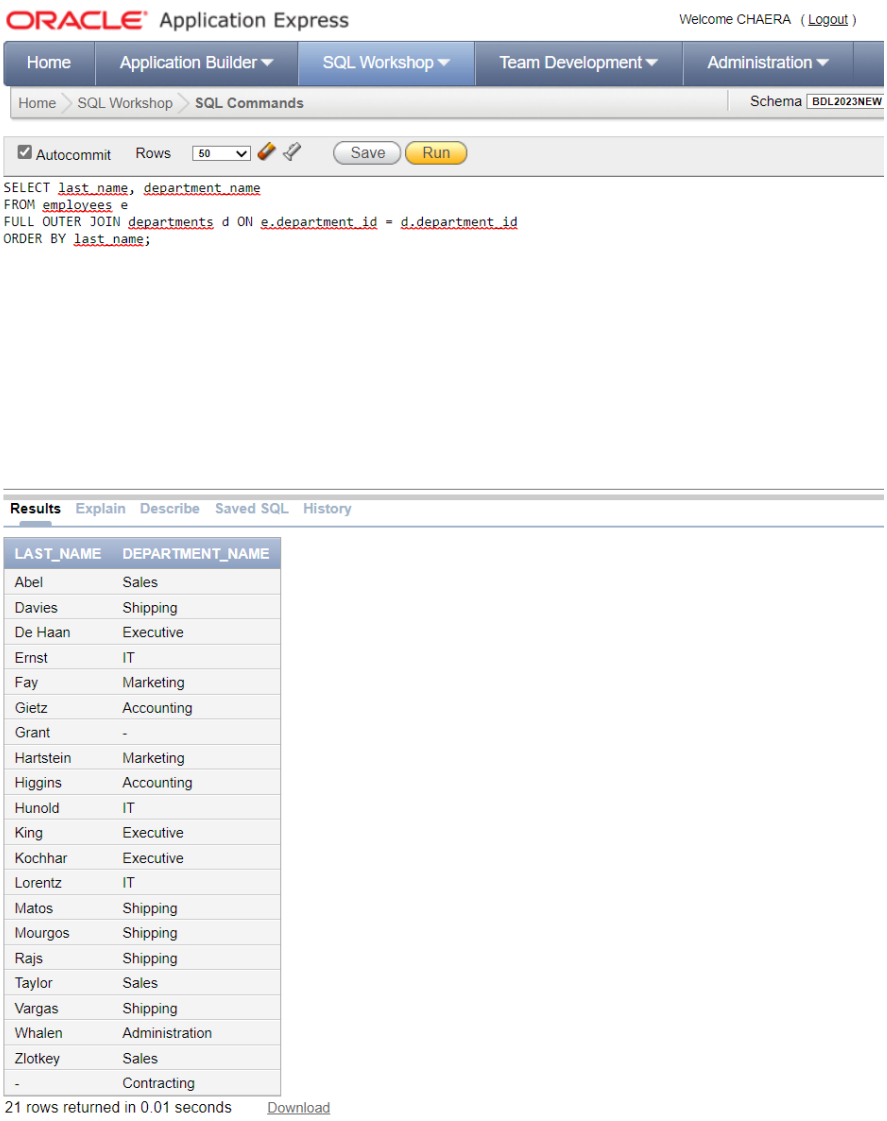


LAST_NAME	SALARY	GRADE_LEVEL
Lorentz	4200	B
Ernst	6000	C
Gietz	8300	C
Taylor	8600	C
Hunold	9000	C
Zlotkey	10500	D
Higgins	12000	D
Abel	11000	D
De Haan	17000	E
Kochhar	17000	E
King	24000	E

15. 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 :



LAST_NAME	DEPARTMENT_NAME
Abel	Sales
Davies	Shipping
De Haan	Executive
Ernst	IT
Fay	Marketing
Gietz	Accounting
Grant	-
Hartstein	Marketing
Higgins	Accounting
Hunold	IT
King	Executive
Kochhar	Executive
Lorentz	IT
Matos	Shipping
Mourgos	Shipping
Rajs	Shipping
Taylor	Sales
Vargas	Shipping
Whalen	Administration
Zlotkey	Sales
-	Contracting

16. 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.

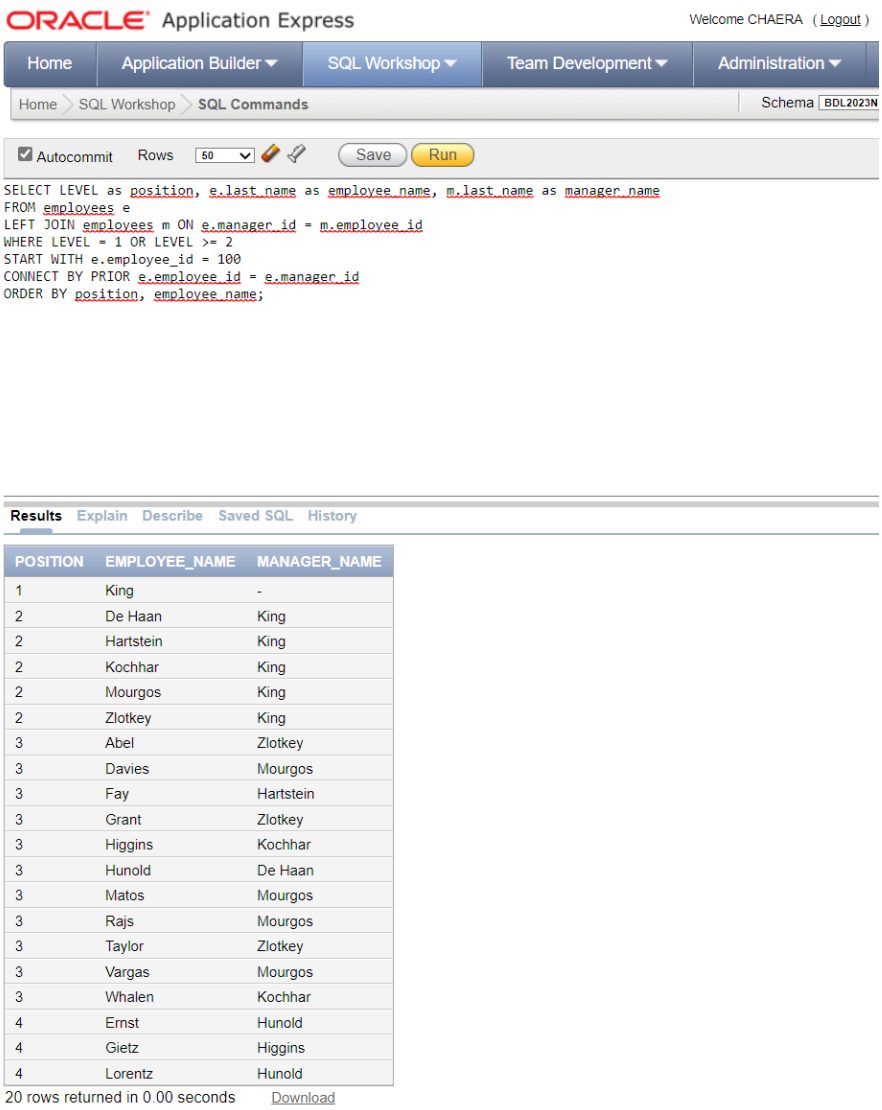
    Start the listing with employee number 100.

    → `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 :

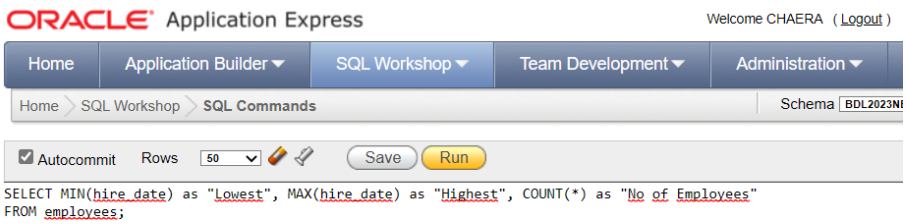


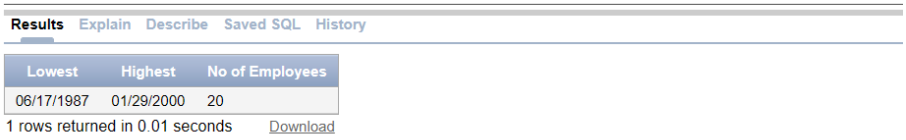
POSITION	EMPLOYEE_NAME	MANAGER_NAME
1	King	-
2	De Haan	King
2	Hartstein	King
2	Kochhar	King
2	Mourgos	King
2	Zlotkey	King
3	Abel	Zlotkey
3	Davies	Mourgos
3	Fay	Hartstein
3	Grant	Zlotkey
3	Higgins	Kochhar
3	Hunold	De Haan
3	Matos	Mourgos
3	Rajs	Mourgos
3	Taylor	Zlotkey
3	Vargas	Mourgos
3	Whalen	Kochhar
4	Ernst	Hunold
4	Gietz	Higgins
4	Lorentz	Hunold

17. 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 :



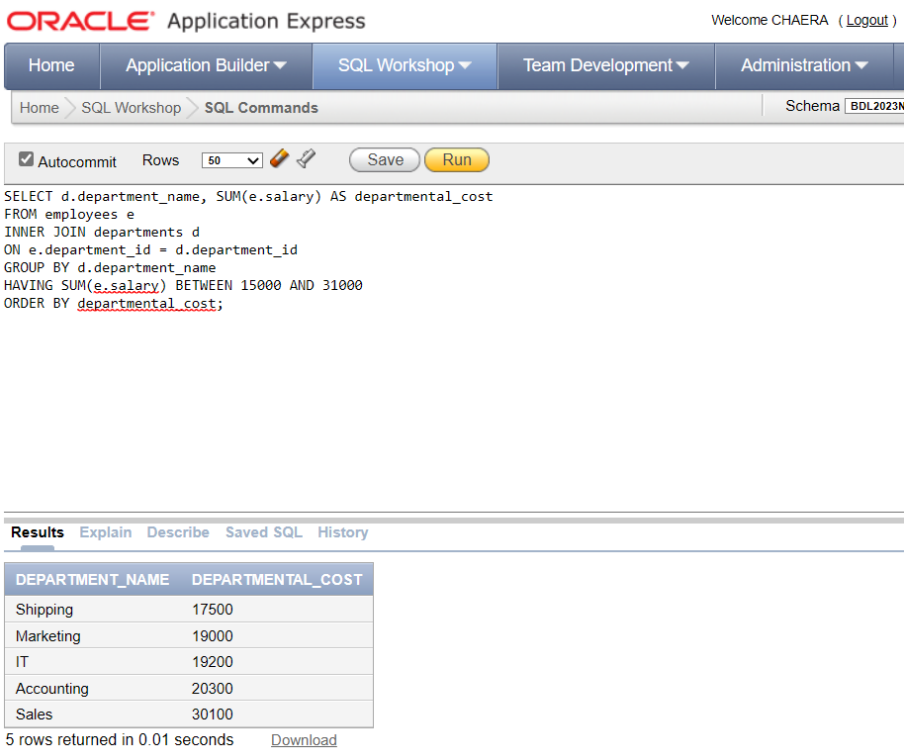


18. Create a list of department names and the departmental costs (salaries added up). Include only departments whose salary costs are between 15000 and 31000, and sort the listing by the cost.

→
```
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 :

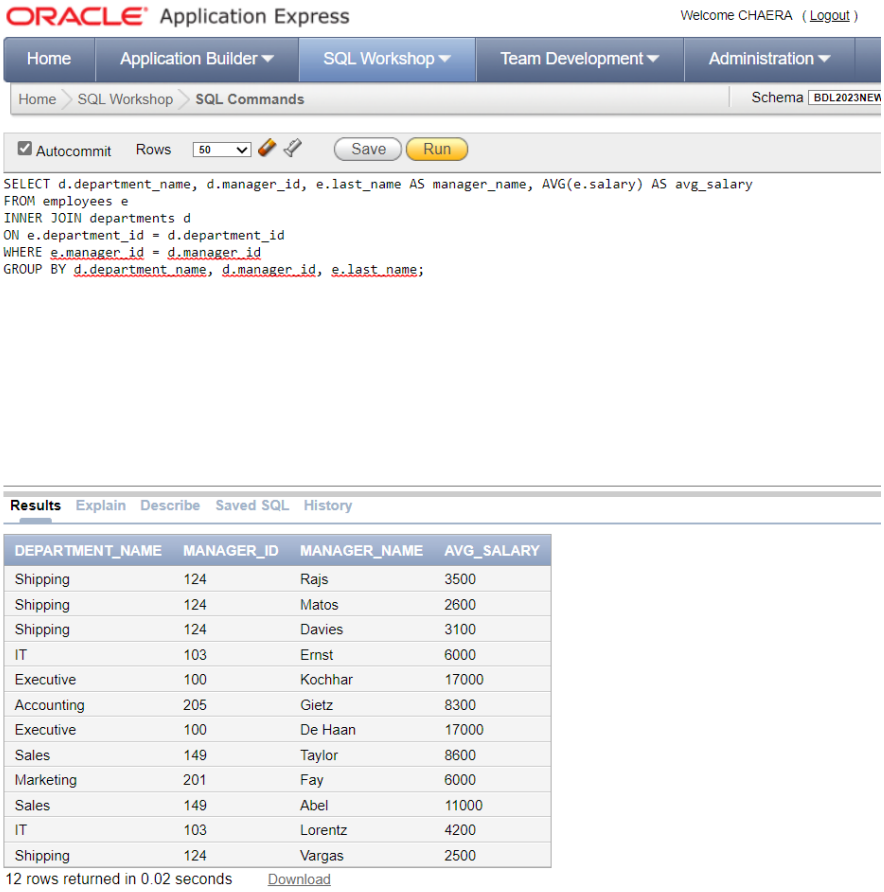


19. Create a list of department names, the manager id, manager name (employee last name) of that department, and the average salary in each department.

→

```
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 :

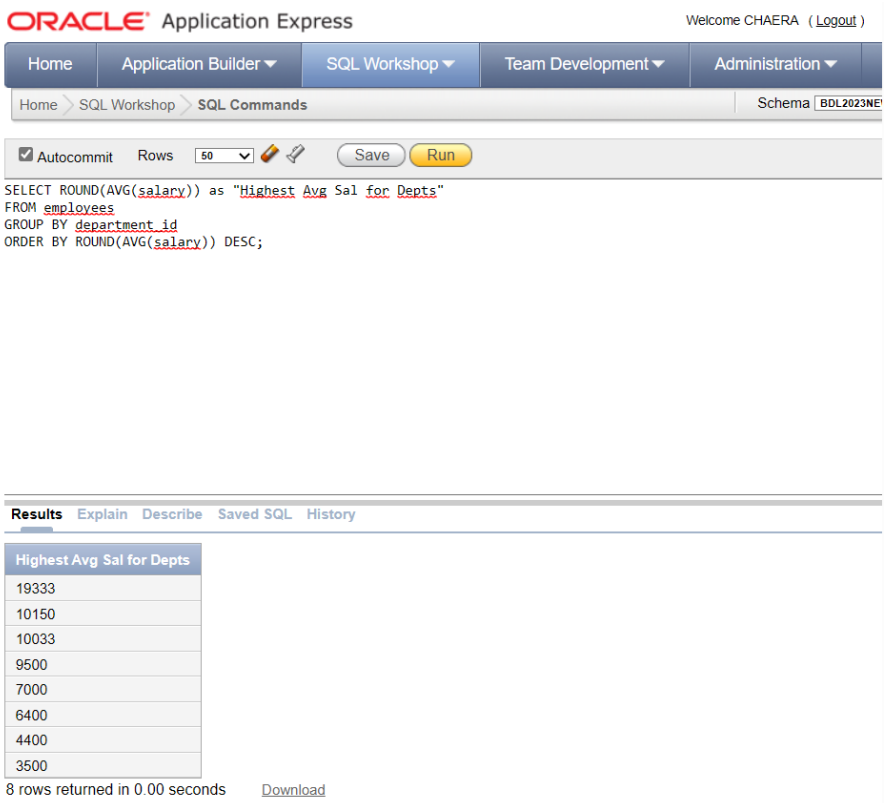


DEPARTMENT_NAME	MANAGER_ID	MANAGER_NAME	AVG_SALARY
Shipping	124	Rajs	3500
Shipping	124	Matos	2600
Shipping	124	Davies	3100
IT	103	Ernst	6000
Executive	100	Kochhar	17000
Accounting	205	Gietz	8300
Executive	100	De Haan	17000
Sales	149	Taylor	8600
Marketing	201	Fay	6000
Sales	149	Abel	11000
IT	103	Lorentz	4200
Shipping	124	Vargas	2500

20. 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 :

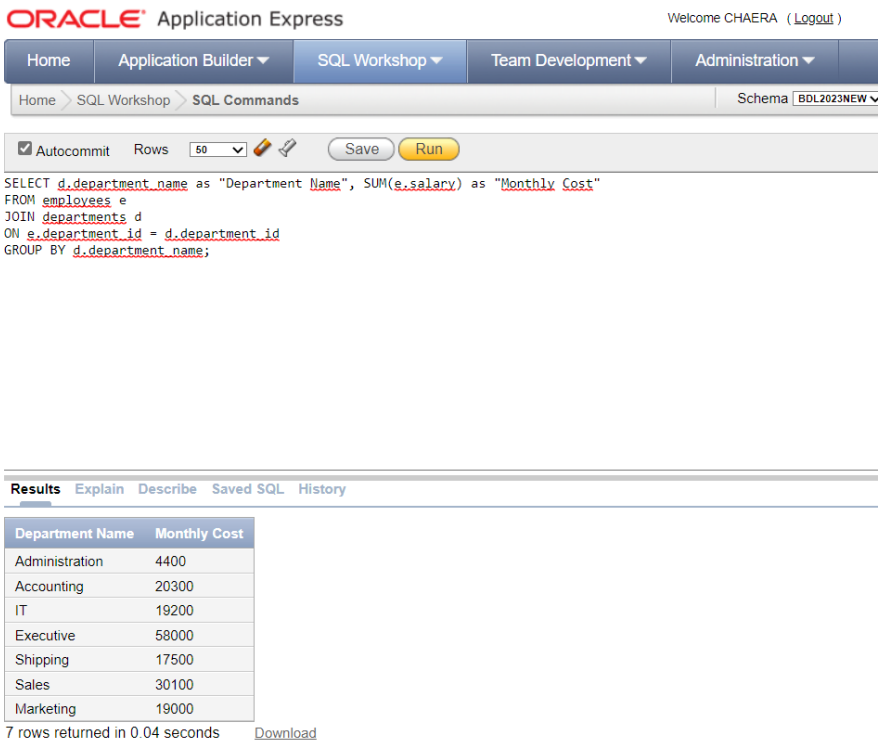


21. Create a list of department names and their monthly costs (salaries added up).

→

```
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 :



22. 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 :

ORACLE Application Express

Welcome CHAERA (Logout)

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > SQL Commands

Schema BDL2023NE

Autocommit Rows 50 Save Run

```
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;
```

Results Explain Describe Saved SQL History

Department Name	Job Title	Monthly Cost
Accounting	AC_ACCOUNT	8300
Accounting	AC_MGR	12000
Accounting	-	20300
Administration	AD_ASST	4400
Administration	-	4400
Executive	AD_PRES	24000
Executive	AD_VP	34000
Executive	-	58000
IT	IT_PROG	19200
IT	-	19200
Marketing	MK_MAN	13000
Marketing	MK_REP	6000
Marketing	-	19000
Sales	SA_MAN	10500
Sales	SA_REP	19600
Sales	-	30100
Shipping	ST_CLERK	11700
Shipping	ST_MAN	5800
Shipping	-	17500

19 rows returned in 0.01 seconds Download

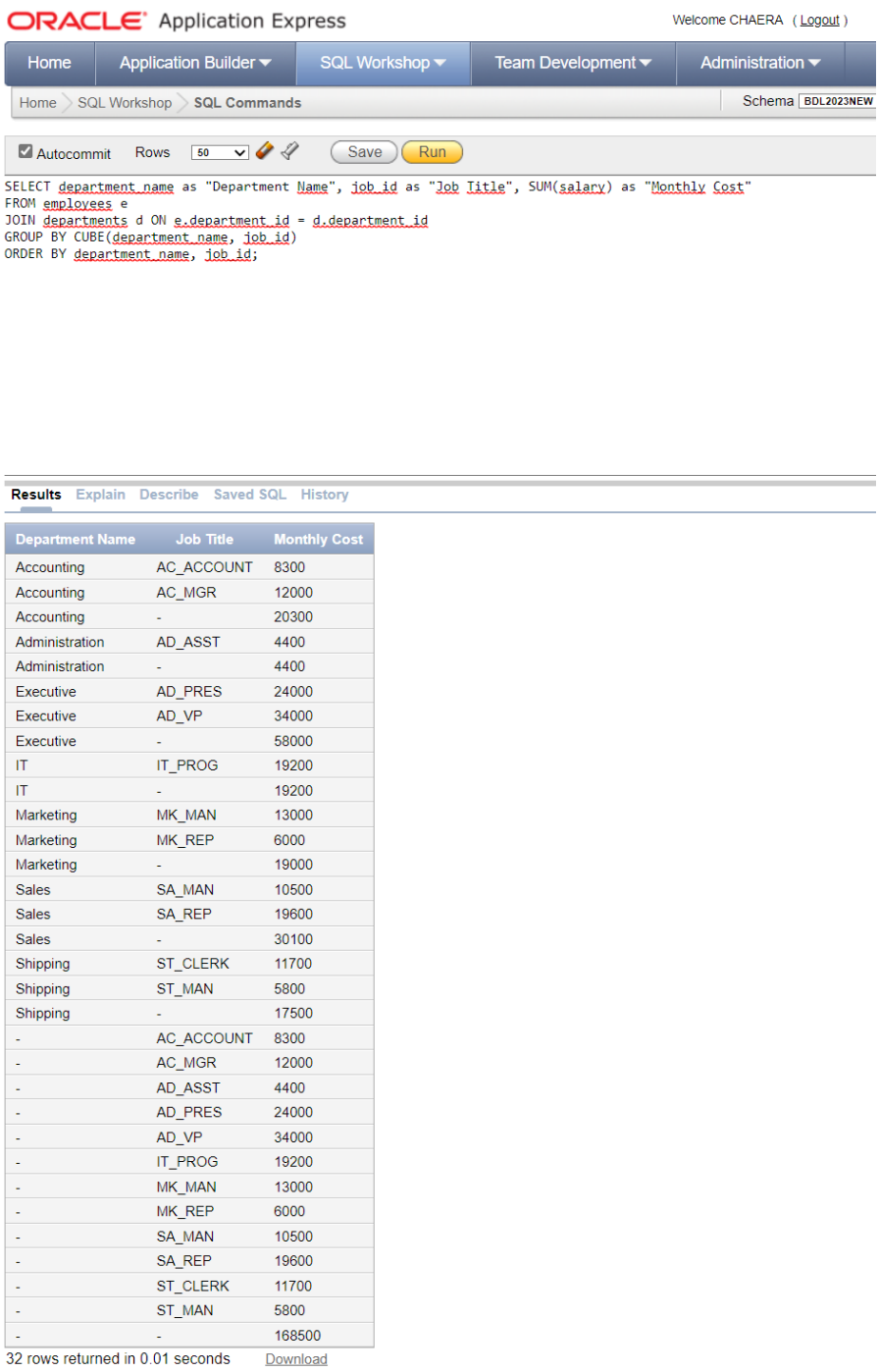
23. Create a list of department names, and job_ids. Calculate the monthly salary cost for each job_id within a department, for each department, for each group of job_ids irrespective of the department, and for all departments added together (Hint: Cube).

→ `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 :



Department Name	Job Title	Monthly Cost
Accounting	AC_ACCOUNT	8300
Accounting	AC_MGR	12000
Accounting	-	20300
Administration	AD_ASST	4400
Administration	-	4400
Executive	AD_PRES	24000
Executive	AD_VP	34000
Executive	-	58000
IT	IT_PROG	19200
IT	-	19200
Marketing	MK_MAN	13000
Marketing	MK_REP	6000
Marketing	-	19000
Sales	SA_MAN	10500
Sales	SA_REP	19600
Sales	-	30100
Shipping	ST_CLERK	11700
Shipping	ST_MAN	5800
Shipping	-	17500
-	AC_ACCOUNT	8300
-	AC_MGR	12000
-	AD_ASST	4400
-	AD_PRES	24000
-	AD_VP	34000
-	IT_PROG	19200
-	MK_MAN	13000
-	MK_REP	6000
-	SA_MAN	10500
-	SA_REP	19600
-	ST_CLERK	11700
-	ST_MAN	5800
-	-	168500

24. 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 :

ORACLE Application Express

Welcome CHAERA (Logout

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > **SQL Commands**

Schema BDL20

Autocommit Rows 50 Save Run

```
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;
```

Results Explain Describe Saved SQL History

Department Name	Job Title	Monthly Cost	DEPARTMENT_ID_USED	JOB_ID_USED
Accounting	AC_ACCOUNT	8300	Yes	No
Accounting	AC_ACCOUNT	8300	Yes	No
Accounting	AC_ACCOUNT	8300	Yes	No
Accounting	AC_ACCOUNT	8300	No	No
Accounting	AC_ACCOUNT	8300	No	No
Accounting	AC_ACCOUNT	8300	No	No
Accounting	AC_MGR	12000	No	No
Accounting	AC_MGR	12000	No	No
Accounting	AC_MGR	12000	Yes	No
Accounting	AC_MGR	12000	No	No
Accounting	AC_MGR	12000	Yes	No
Accounting	AC_MGR	12000	Yes	No
Accounting	-	20300	Yes	Yes
Accounting	-	20300	No	Yes
Administration	AD_ASST	4400	Yes	No
Administration	AD_ASST	4400	No	No
Administration	AD_ASST	4400	No	No
Administration	AD_ASST	4400	Yes	No
Administration	AD_ASST	4400	No	No
Administration	AD_ASST	4400	Yes	No
Administration	-	4400	Yes	Yes
Administration	-	4400	No	Yes
Executive	AD_PRES	24000	Yes	No
Executive	AD_PRES	24000	No	No
Executive	AD_PRES	24000	Yes	No
Executive	AD_PRES	24000	No	No
Executive	AD_PRES	24000	Yes	No
Executive	AD_PRES	24000	No	No
Executive	AD_VP	34000	No	No
Executive	AD_VP	34000	No	No
Executive	AD_VP	34000	Yes	No
Executive	AD_VP	34000	No	No
Executive	AD_VP	34000	Yes	No
Executive	AD_VP	34000	Yes	No
Executive	-	58000	No	Yes
Executive	-	58000	Yes	Yes
IT	IT_PROG	19200	No	No
IT	IT_PROG	19200	Yes	No
IT	IT_PROG	19200	No	No
IT	IT_PROG	19200	Yes	No
IT	IT_PROG	19200	No	No
IT	IT_PROG	19200	Yes	No
IT	-	19200	Yes	Yes

25. 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 :

ORACLE Application Express

Welcome CHAERA (Logout)

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > SQL Commands

Schema BDL2023NE

Autocommit Rows 50 Save Run

```
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;
```

Results Explain Describe Saved SQL History

DEPARTMENT_NAME	JOB_ID	CITY	SUM(SALARY)
Accounting	AC_ACCOUNT	-	8300
Accounting	AC_MGR	-	12000
Administration	AD_ASST	-	4400
Executive	AD_PRES	-	24000
Executive	AD_VP	-	34000
IT	IT_PROG	-	19200
Marketing	MK_MAN	-	13000
Marketing	MK_REP	-	6000
Sales	SA_MAN	-	10500
Sales	SA_REP	-	19600
Shipping	ST_CLERK	-	11700
Shipping	ST_MAN	-	5800
-	AC_ACCOUNT	Seattle	8300
-	AC_MGR	Seattle	12000
-	AD_ASST	Seattle	4400
-	AD_PRES	Seattle	24000
-	AD_VP	Seattle	34000
-	IT_PROG	Southlake	19200
-	MK_MAN	Toronto	13000
-	MK_REP	Toronto	6000
-	SA_MAN	Oxford	10500
-	SA_REP	Oxford	19600
-	ST_CLERK	South San Francisco	11700
-	ST_MAN	South San Francisco	5800

24 rows returned in 0.11 seconds Download

26. 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 :

ORACLE Application Express

Welcome CHAERA (Logout)

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > SQL Commands

Schema BDL2023NEW

Autocommit Rows 50 Save Run

```
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;
```

Results Explain Describe Saved SQL History

Employee ID	Employee Name	Department ID	Department Name	City
200	Jennifer Whalen	10	-	-
-	-	10	Administration	-
201	Michael Hartstein	20	-	-
202	Pat Fay	20	-	-
-	-	20	Marketing	-
124	Kevin Mourgos	50	-	-
141	Trenna Rajs	50	-	-
142	Curtis Davies	50	-	-
143	Randall Matos	50	-	-
144	Peter Vargas	50	-	-
-	-	50	Shipping	-
103	Alexander Hunold	60	-	-
104	Bruce Ernst	60	-	-
107	Diana Lorentz	60	-	-
-	-	60	IT	-
149	Eleni Zlotkey	80	-	-
174	Ellen Abel	80	-	-
176	Jonathon Taylor	80	-	-
-	-	80	Sales	-
100	Steven King	90	-	-
101	Neena Kochhar	90	-	-
102	Lex De Haan	90	-	-
-	-	90	Executive	-
205	Shelley Higgins	110	-	-
206	William Gietz	110	-	-
-	-	110	Accounting	-
-	-	190	Contracting	-
178	Kimberely Grant	-	-	-
-	-	-	-	Oxford
-	-	-	-	Seattle
-	-	-	-	South San Francisco
-	-	-	-	Southlake
-	-	-	-	Toronto

33 rows returned in 0.01 seconds Download

27. 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 :