INFO 6210
Data Management and Database Design
In Class Exam Solutions

Professor: Nik Bear Brown

Exam is Tuesday April 9, 2019
One page cheat sheet. No books, computer or phone.

All are 5 points unless noted otherwise.

1 (10 points)) Write a query in SQL to display the employee ID, name, salary, department name, location, department ID, job name of all the employees working at SYDNEY or working in the FINANCE department with an annual salary above 28000, but the monthly salary should not be 3000 or 2800 and who does not works as a MANAGER and whose ID containing a digit of '3' or '7' in 3rd position. List the result in ascending order of department ID and descending order of job name.

Sample table: employees

```
 emp_id | emp_name | job_name  | manager_id | hire_date  | salary  | commission |
dep_id
```

Sample table: department

```
 dep_id |  dep_name  | dep_location
```

Solution:

```
SELECT E.emp_id,
       E.emp_name,
       E.salary,
       D.dep_name,
       D.dep_location,
       E.dep_id,
       E.job_name
FROM employees E,
     department D
WHERE (D.dep_location = 'SYDNEY'
       OR D.dep_name = 'FINANCE')
  AND E.dep_id=D.dep_id
  AND E.emp_id IN
    (SELECT emp_id
     FROM employees E
     WHERE (12*E.salary) > 28000
       AND E.salary NOT IN (3000,
                            2800)
       AND E.job_name !='MANAGER'
       AND (trim(to_char(emp_id,'99999')) LIKE '__3%'
            OR trim(to_char(emp_id,'99999')) LIKE '__7%'))
ORDER BY E.dep_id ASC,
         E.job_name DESC;
```

2 (10 points) Write a query in SQL to list all the employees of grade 2 and 3.

Sample table: employees

```
 emp_id | emp_name | job_name  | manager_id | hire_date  | salary  | commission |
dep_id
```

Sample table: salary_grade

```
 grade | min_sal | max_sal
```

Solution:

```
SELECT *
FROM employees e,
     salary_grade s
WHERE e.salary BETWEEN s.min_sal AND s.max_sal
  AND s.grade IN (2, 3);
```

3 (10 points) Write a query in SQL to find the name of the venue with city where the EURO cup 2016 final match was played

Sample table: soccer_venue

```
 venue_id |       venue_name        | city_id | aud_capacity
----------+-------------------------+---------+--------------
```

Sample table: soccer_city

```
 city_id |     city      | country_id
---------+---------------+------------
```

Sample table: match_mast

```
 match_no | play_stage | play_date  | results | decided_by | goal_score | venue_id |
referee_id | audence | plr_of_match | stop1_sec | stop2_sec
----------+------------+------------+---------+------------+------------+----------+--
----------+---------+--------------+-----------+-----------
```

Solution:

```
SELECT venue_name, city
FROM soccer_venue a
JOIN soccer_city b ON a.city_id=b.city_id
JOIN match_mast d ON d.venue_id=a.venue_id
AND d.play_stage='F';
```

4) Write a query in SQL to find the number of goal scored by each team in every match within normal play schedule

Sample table: match_details

```
 match_no | play_stage | team_id | win_lose | decided_by | goal_score | penalty_score
| ass_ref | player_gk
----------+------------+---------+----------+------------+------------+---------------
+---------+-----------
```

Sample table: soccer_country

```
 country_id | country_abbr |    country_name
------------+--------------+--------------------
```

Solution:

```
SELECT match_no,country_name,goal_score
FROM match_details a
JOIN soccer_country b
ON a.team_id=b.country_id
WHERE decided_by='Normal play schedule'
ORDER BY match_no;
```

5) Write a query in SQL to find the highest individual scorer in EURO cup 2016.

Sample table: goal_details

```
 goal_id | match_no | player_id | team_id | goal_time | goal_type | play_stage |
goal_schedule | goal_half
---------+----------+-----------+---------+-----------+-----------+------------+------
---------+-----------
```

Sample table: player_mast

```
 player_id | team_id | jersey_no |       player_name       | posi_to_play | dt_of_bir
| age |    playing_club
```

```
----------+--------+----------+-------------------------+-------------+----------
-+-----+--------------------

Sample table: soccer_country

 country_id | country_abbr |    country_name
------------+--------------+--------------------
```

Solution:

```
SELECT player_name,country_name,count(player_name)
FROM goal_details gd
JOIN player_mast pm ON gd.player_id =pm.player_id
JOIN soccer_country sc ON pm.team_id = sc.country_id
GROUP BY country_name,player_name HAVING COUNT(player_name) >= ALL
  (SELECT COUNT(player_name)
   FROM goal_details gd
   JOIN player_mast pm ON gd.player_id =pm.player_id
   JOIN soccer_country sc ON pm.team_id = sc.country_id
   GROUP BY country_name,player_name);
```

6) Write a query in SQL to find the name and country of the referee who assisted the referee in the final match.

```
Sample table: asst_referee_mast

ass_ref_id |       ass_ref_name       | country_id
------------+--------------------------+------------

Sample table: soccer_country

country_id | country_abbr |    country_name
------------+--------------+--------------------

Sample table: match_details

match_no | play_stage | team_id | win_lose | decided_by | goal_score |
penalty_score | ass_ref | player_gk
----------+------------+---------+----------+------------+------------+------
---------+---------+-----------
```

Solution:

```
SELECT ass_ref_name, country_name
FROM asst_referee_mast a
JOIN soccer_country b
ON a.country_id=b.country_id
JOIN match_details c
ON a.ass_ref_id=c.ass_ref
WHERE play_stage='F';
```

7) You have two SQL tables. The first one is called employees and it contains the employee names, the unique employee ids and the department names of a company.

Sample:
department_name employee_id employee_name

The second one is named salaries. It holds the same employee names and the same employee ids ñ and the salaries for each employee.

Sample:
salary employee_id employee_name

The company has 546 employees, so both tables have 546 rows.
Print every department where the average salary per employee is lower than $500.

Solution:

```
SELECT department_name, AVG(salaries.salary) AS avg_salaries
FROM employees
JOIN salaries
ON employees.employee_id = salaries.employee_id
GROUP BY department_name
```

8) You have two SQL tables: authors and books.

authors:
author_name book_name

books:
book_name sold_copies

Create an SQL query that shows the TOP 3 authors who sold the most books in total.

Solution:

```
SELECT authors.author_name, SUM(books.sold_copies) AS sold_sum
FROM authors
JOIN books
ON books.book_name = authors.book_name
GROUP BY authors.author_name
ORDER BY sold_sum DESC
```

```
LIMIT 3;
```

9) Write a SQL statement to make a list in ascending order for the customer who works either through a salesman or by own.

Sample table: customer

customer_id	cust_name	city	grade	salesman_id

Sample table: salesman

salesman_id	name	city	commission

Questions 10 through 14 use the following tables employees and department.

Sample table: employees

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id

Sample table: department

dep_id	dep_name	dep_location

Solution:

```
SELECT a.cust_name,a.city,a.grade,
b.name AS "Salesman",b.city
FROM customer a
LEFT JOIN salesman b
ON a.salesman_id=b.salesman_id
order by a.customer_id;
```

10. Group average salary by job_name

Solution:

```
SELECT job_name,
       avg(salary)
FROM employees
GROUP BY job_name;
```

11. Count the number of unique job names.

Solution:

```
SELECT COUNT(DISTINCT job_name)
FROM employee;
```

12. Assume 10% tax rate. Create a column that is the salary after taxes.

Solution:

```
ALTER TABLE employee
ADD Total INT NOT NULL DEFAULT 0

INSERT INTO employee(Total)
SELECT 0.9*salary FROM employee;
```

13. Count all of employee ids that contain a job name the word 'data' in it.

Solution:

```
SELECT COUNT(emp_id)
FROM employees
WHERE job_name LIKE '%data%';
```

14. Subselect columns using a subquery.

Solution:

Subquery to Calculate Average salary –

```
SELECT emp_id,
emp_name,
(SELECT AVG(salary)
   FROM employee) AS AverageSalary
FROM   employee;
```

14. Why create views?

Solution:

1) Views are used for security purposes because they provide encapsulation of the name of the table. Data is in the virtual table, not stored permanently. Views display only selected data.

```
The syntax for creating a View is given below:
Create View Viewname As
 Select Column1, Column2  From Tablename
 Where (Condition) Group by (Grouping Condition) having (having Condition)
```

2) Views can hide complexity

If you have a query that requires joining several tables, or has complex logic or calculations, you can code all that logic into a view, then select from the view just like you would a table.

3) Views can be used as a security mechanism

A view can select certain columns and/or rows from a table, and permissions set on the view instead of the underlying tables. This allows surfacing only the data that a user needs to see.

4) Views can simplify supporting legacy code

If you need to refactor a table that would break a lot of code, you can replace the table with a view of the same name. The view provides the exact same schema as the original table, while the actual schema has changed. This keeps the legacy code that references the table from breaking, allowing you to change the legacy code at your leisure.

15. Explain to an eight year old (i.e. your professor) what are the first three Normal Forms.

Solution:

First normal form (1NF)
• Each table has a primary key: minimal set of attributes which can uniquely identify a
record

• The values in each column of a table are atomic (No multi-value attributes allowed).
2
• There are no repeating groups: two columns do not store similar information in the
same table.
Second normal form (2NF)
• All requirements for 1st NF must be met.
• No partial dependencies.
• No calculated data
Third normal form (3NF)
• All requirements for 2nd NF must be met.
• Eliminate fields that do not directly depend on the primary key; that is no transitive
dependencies.

Explain above using example.

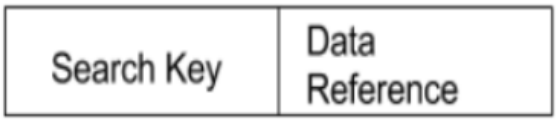
16. What is an index? Why do we use them?

Solution:

An index or database index is a data structure which is used to quickly locate and access the data in a database table. It is used to speed up queries.

Indexes are created using some database columns.

The first column is the Search key that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly (Note that the data may or may not be stored in sorted order)
.
The second column is the Data Reference which contains a set of pointers holding the address of the disk block where that particular key value can be found



Structure of an index

Uses of Indexes -

1) Support for fast lookup - Indexing is a way to optimize performance of a database by minimizing the number of disk accesses required when a query is processed.
2) Policing the database constraints - Indexes are used to police database constraints, such as UNIQUE, EXCLUSION, PRIMARY KEY and FOREIGN KEY. An

index may be declared as UNIQUE, which creates an implicit constraint on the underlying table. Database systems usually implicitly create an index on a set of columns declared PRIMARY KEY.

Many database systems require that both referencing and referenced sets of columns in a FOREIGN KEY constraint are indexed, thus improving performance of inserts, updates and deletes to the tables participating in the constraint.