Database Programming with SQL

7-1: Oracle Equijoin and Cartesian Product

1. Create a Cartesian product that displays the columns in the d_play_list_items and the d_track_listings in the DJs on Demand database.

```
1  SELECT *
2  FROM d_play_list_items, d_track_listings;
```

Results | Explain | Describe | Saved SQL | History

EVENT_ID	SONG_ID	COMMENTS	SONG_ID	CD_NUMBER	TRACK
100	45	Play late	45	92	1
100	46	-	45	92	1
100	47	Play early	45	92	1
105	48	Play after cake cutting	45	92	1
105	49	Play first	45	92	1
105	47	Play for the father	45	92	1
100	45	Play late	46	93	1
100	46	-	46	93	1
100	47	Play early	46	93	1
105	48	Play after cake cutting	46	93	1

2. Correct the Cartesian product produced in question 1 by creating an equijoin using a common column.

```
1  SELECT DPI.*, DTL.*
2  FROM d_play_list_items DPI
3  JOIN d_track_listings DTL ON DPI.SONG_ID = DTL.SONG_ID;
```

Results | Explain | Describe | Saved SQL | History

EVENT_ID	SONG_ID	COMMENTS	SONG_ID	CD_NUMBER	TRACK
100	45	Play late	45	92	1
100	46	-	46	93	1
105	47	Play for the father	47	91	2
100	47	Play early	47	91	2
105	48	Play after cake cutting	48	95	5
105	49	Play first	49	91	3

3. Write a query to display the title, type, description, and artist from the DJs on Demand database.

```
SELECT CDS.TITLE, CDS.PRODUCER
FROM D_TRACK_LISTINGS T
JOIN D_CDS CDS ON T.CD_NUMBER = CDS.CD_NUMBER;
```

Results | Explain | Describe | Saved SQL | History

TITLE	PRODUCER
Back to the Shire	Middle Earth Records
Songs from My Childhood	Old Town Records
Party Music for All Occasions	The Music Man
Here Comes the Bride	The Music Man
Party Music for All Occasions	The Music Man

The D_CDS table doesn't have columns type and description, so only title and producer have been included in this query.

4. Rewrite the query in question 3 to select only those titles with an ID of 47 or 48.

No title with ID's 47 and 48 exist.

5. Write a query that extracts information from three tables in the DJs on Demand database, the d_clients table, the d_events table, and the d_job_assignments table.

```
SELECT c.client_number, c.first_name, c.last_name, e.event_date, e.name AS event_name, ja.job_date, ja.status
FROM d_clients c
JOIN d_events e ON c.client_number = e.client_number
JOIN d_job_assignments ja ON e.id = ja.event_id;
```

Results | Explain | Describe | Saved SQL | History

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	EVENT_DATE	EVENT_NAME	JOB_DATE	STATUS
6133	Lauren	Vigil	28-Apr-2004	Vigil wedding	02-Feb-2004	Visited

6. Create and execute an equijoin between DJs on Demand tables d_track_listings and d_cds. Return the song_id and the title only.

```
SELECT tl.song_id, c.title
FROM d_track_listings tl
JOIN d_cds c ON tl.cd_number = c.cd_number;
```

Results | Explain | Describe | Saved SQL | History

SONG_ID	TITLE
45	Back to the Shire
48	Here Comes the Bride
47	Party Music for All Occasions
49	Party Music for All Occasions
46	Songs from My Childhood

7. Mark T for the statements that are true and F for the statements that are false.

__**T**__ a. A join is a type of query that gets data from more than one table based on columns with the same name.
__**T**__ b. To join tables using an equijoin, there must be a common column in both tables and that column is usually a primary key in one of the tables.
__**T**__ c. A Cartesian product occurs because the query does not specify a WHERE clause.
__**F**__ d. Table aliases are required to create a join condition.
__**T**__ e. If a table alias is used for a table name in the FROM clause, it must be substituted for the table name throughout the SELECT statement.
__**F**__ f. Table alias must be only one character in length.
__**T**__ g. A simple join or inner join is the same as an equijoin.

8. What advantage does being able to combine data from multiple tables have for a business?
    - **Comprehensive Insights**: Businesses can gain a more holistic view of their operations, customers, and events by combining data from different sources.
    - **Enhanced Decision-Making**: By analyzing data across multiple tables, businesses can make more informed decisions, leading to better strategies and outcomes.
    - **Efficiency in Data Management**: Combining data reduces redundancy and improves data integrity, leading to a more streamlined database management process.
    - **Customized Reports**: Businesses can create detailed reports that cater to specific needs by merging relevant data from various tables.
    - **Improved Customer Service**: By analyzing customer interactions and event details together, businesses can enhance their service offerings and customer satisfaction.

7-1: Oracle Equijoin and Cartesian Product

1. Create a join based on the cost of the event between the DJs on Demand tables D_EVENTS and D_PACKAGES. Show the name of the event and the code for each event.

```
1  SELECT e.name AS event_name, e.package_code
2  FROM d_events e
3  JOIN d_packages p ON e.cost BETWEEN p.low_range AND p.high_range;
```

Results | Explain | Describe | Saved SQL | History

EVENT_NAME	PACKAGE_CODE
Vigil wedding	200
Peters Graduation	112

2. Using the Oracle database, create a query that returns the employee last name, salary, and job_grade level based on the salary. Select the salary between the lowest and highest salaries.

```
1  SELECT e.last_name, e.salary, g.grade_level
2  FROM employees e
3  JOIN job_grades g ON e.salary BETWEEN g.lowest_sal AND g.highest_sal;
```

Results | Explain | Describe | Saved SQL | History

LAST_NAME	SALARY	GRADE_LEVEL
Vargas	2500	A
Matos	2600	A
Heiden	2600	A
Davies	3100	B
Rajs	3500	B
Bell	3500	B
Stocks	3700	B
Fay	3900	B
TAYLOR	4000	B
Ricci	4100	B

3. What condition requires the creation of a nonequijoin?
    - When equality isn't involved between two tables during join condition, a **nonequijoin** is required. For example, when comparing ranges, a nonequijoin is used.

4. Rewrite the following nonequijoin statement using the logical condition operators (AND, OR, NOT): WHERE a.ranking BETWEEN g.lowest_rank AND g.highest_rank
    - WHERE a.ranking >= g.lowest_rank AND a.ranking <= g.highest_rank;

5. How do you know when to use a table alias and when not to use a table alias?
    - **Use a table alias** when:
        - You are joining multiple tables to avoid ambiguity between column names.
        - You want to shorten the table name for readability.

      Example:

```
1  SELECT e.last_name, d.department_name
2  FROM employees e
3  JOIN departments d ON e.department_id = d.department_id;
```

Results | Explain | Describe | Saved SQL | History

LAST_NAME	DEPARTMENT_NAME
Ricci	Administration
Saikawa	Administration
Hernandez	Administration
Whalen	Administration
Safwah	Marketing
Newton	Marketing
Steiner	Marketing
Stocks	Marketing
TAYLOR	Marketing
Hartstein	Marketing

    - Do not use a table alias when:
        - There is only one table or when aliases would not improve the clarity of your SQL.

6. What kind of join would you use if you wanted to find data between a range of numbers?
    - To find data between a range of numbers, a **nonequijoin** is appropriate. This is because the condition for joining the tables is based on a range rather than equality.

7. You need to produce a report for Global Fast Foods showing customers and orders. A customer must be included on the report even if the customer has had no orders.

```
1  SELECT e.last_name, d.department_name
2  FROM employees e
3  JOIN departments d ON e.department_id = d.department_id;
```

Results | Explain | Describe | Saved SQL | History

LAST_NAME	DEPARTMENT_NAME
Ricci	Administration
Saikawa	Administration
Hernandez	Administration
Whalen	Administration
Safwah	Marketing
Newton	Marketing
Steiner	Marketing
Stocks	Marketing
TAYLOR	Marketing
Hartstein	Marketing