Assignment-5

Name- Ajay Chaudhary
Batch- Data Engineering (Batch-1)

Data cleaning & Transformation queries

Table for data cleaning

```
create table studentdata1(
id int,
name varchar(250),
age int,
grade varchar(20)
);
select * from studentdata1;
insert into studentdata1(id,name,age,grade) values(null,'stella',20,'A+');
insert into studentdata1(id,name,age,grade) values(1,'appu',20,'A+');
insert into studentdata1(id,name,age,grade) values(5,'bob',21,'C');
insert into studentdata1(id,name,age,grade) values(6,'sunny',21,null);
insert into studentdata1(id,name,age,grade) values(7,null,21,'C');
```

Perfoming data cleaning

STEP-1 ----> Deleting the duplicate data

```
select name,count(name) as Actual_count from studentdata1
group by name
having count(name)>1;
```

100% 1:22

Result Grid Filter Rows: Search Export:

name	Actual_count

STEP-2 ---> removing the null values over here

```
select * from studentdata1 where id is null;
```

100% 1:26

Result Grid Filter Rows: Search Export:

id	name	age	grade
NULL	stella	20	A+

selecting the data where student name is null

```
31 •  select * from studentdata1
32    where name is null;
```

100% 20:32

Result Grid | Filter Rows: Search | Expo

id	name	age	grade
7	NULL	21	C

```
34 •  delete from studentdata1
35    where name is null;
36
```

100% 1:33

Action Output

	Time	Action
19	22:46:15	delete from studentdata1 where name is null

updating the null values where id and grade column is null

```
39 •  select * from studentdata1
40    where id is null;
```

100% 18:40

Result Grid | Filter Rows: Search | Export:

id	name	age	grade
NULL	stella	20	A+

```
41
42 •  update studentdata1 set id=7
43    where id is null;
```

100% 1:41

Action Output

	Time	Action
21	22:47:39	update studentdata1 set id=7 where id is null

id	name	age	grade
1	appu	20	A+
5	bob	21	C
6	sunny	21	NULL
7	stella	20	A+

update the value of grade also

```
49  update studentdata1 set grade='A'
50  where grade is null;
51
```

100% 1:49

Action Output

	Time	Action
23	22:48:59	update studentdata1 set grade='A' where grade is null

id	name	age	grade
1	appu	20	A+
5	bob	21	C
6	sunny	21	A
7	stella	20	A+

Capitalization of names in data

```
53  update  studentdata1
54  set name=UPPER(name);
55
```

100% 1:53

ction Output

	Time	Action
25	22:50:07	update studentdata1 set name=UPPER(name)

id	name	age	grade
1	APPU	20	A+
5	BOB	21	C
6	SUNNY	21	A
7	STELLA	20	A+

Ranking Functions

```
57  CREATE TABLE ExamResult
58  (
59  StudentName VARCHAR(70),
60   Subject     VARCHAR(20),
61   Marks       INT
62  );
63
64  INSERT INTO ExamResult VALUES('Lily','Maths',65);
65  INSERT INTO ExamResult VALUES('Lily','Science',80);
66  INSERT INTO ExamResult VALUES('Lily', 'english',70);
67  INSERT INTO ExamResult VALUES('Isabella','Maths',50);
68  INSERT INTO ExamResult VALUES('Isabella','Science',70);
69  INSERT INTO ExamResult VALUES('Isabella','english',90);
70  INSERT INTO ExamResult VALUES('Olivia','Maths',55);
71  INSERT INTO ExamResult VALUES('Olivia','Science',60);
72  INSERT INTO ExamResult VALUES('Olivia','english',89);
```

ROW_Number() SQL RANK function

We use ROW_Number() SQL RANK function to get a unique sequential number for each row in the specified data. It gives the rank one for the first row and then increments the value by one for each row. We get different ranks for the row having similar values as well.

```
75  SELECT Studentname, Subject, Marks, ROW_NUMBER() OVER(ORDER BY Marks) RowNumber
76  FROM ExamResult;
```

100% 17:76

Result Grid Filter Rows: Search Export:

Studentname	Subject	Marks	RowNumber
Isabella	Maths	50	1
Olivia	Maths	55	2
Olivia	Science	60	3
Lily	Maths	65	4
Lily	english	70	5
Isabella	Science	70	6
Lily	Science	80	7
Olivia	english	89	8
Isabella	english	90	9

RANK() SQL RANK Function

We use RANK() SQL Rank function to specify rank for each row in the result set.

```sql
SELECT
    StudentName,
    Subject,
    Marks,
    RANK() OVER (PARTITION BY Subject ORDER BY Marks DESC) AS SubjectRank
FROM
    ExamResult;
```

100% 1:79

Result Grid Filter Rows: Search Export:

StudentName	Subject	Marks	SubjectRank
Isabella	english	90	1
Olivia	english	89	2
Lily	english	70	3
Lily	Maths	65	1
Olivia	Maths	55	2
Isabella	Maths	50	3
Lily	Science	80	1
Isabella	Science	70	2
Olivia	Science	60	3

DenseRANK() sql function

We use DENSE_RANK() function to specify a unique rank number within the partition as per the specified column value.

```sql
SELECT
    StudentName,
    Subject,
    Marks,
    DENSE_RANK() OVER (PARTITION BY Subject ORDER BY Marks) AS SubjectDenseRank
FROM
    ExamResult;
```

100% 16:94

Result Grid Filter Rows: Search Export:

StudentName	Subject	Marks	SubjectDenseRank
Lily	english	70	1
Olivia	english	89	2
Isabella	english	90	3
Isabella	Maths	50	1
Olivia	Maths	55	2
Lily	Maths	65	3
Olivia	Science	60	1
Isabella	Science	70	2
Lily	Science	80	3

NTILE(N) SQL RANK function

We use the NTILE(N) function to distribute the number of rows in the specified (N) number of groups. Each row group gets its rank as per the specified condition. We need to specify the value for the desired number of groups.

```
96      SELECT *,
97          NTILE(2) OVER(
98          ORDER BY Marks DESC) as ntiles
99      FROM ExamResult
100     ORDER BY ntiles;
101
```

100% 17:100

Result Grid Filter Rows: Search Export:

StudentName	Subject	Marks	ntiles
Isabella	english	90	1
Olivia	english	89	1
Lily	Science	80	1
Lily	english	70	1
Isabella	Science	70	1
Lily	Maths	65	2
Olivia	Science	60	2
Olivia	Maths	55	2
Isabella	Maths	50	2

Stored Procedures-

Tables used for stored procedures

```
104     CREATE TABLE Product
105     (ProductID INT, ProductName VARCHAR(100) );
106
107     CREATE TABLE ProductDescription
108     (ProductID INT, ProductDescription VARCHAR(800) )
109
110     INSERT INTO Product VALUES (680,'HL Road Frame - Black, 58')
111     ,(706,'HL Road Frame - Red, 58')
112     ,(707,'Sport-100 Helmet, Red')
113
114     INSERT INTO ProductDescription VALUES (680,'Replacement mountain wheel for entry-level rider.')
115     ,(706,'Sturdy alloy features a quick-release hub.')
116     ,(707,'Aerodynamic rims for smooth riding.')
```

```
139  DELIMITER //
140  CREATE PROCEDURE GetProductInfo(IN p_ProductID INT)
141  BEGIN
142      SELECT
143          p.ProductID,
144          p.ProductName,
145          pd.ProductDescription
146      FROM
147          Product p
148      JOIN
149          ProductDescription pd ON p.ProductID = pd.ProductID
150      WHERE
151          p.ProductID = p_ProductID;
152  END //
153
154  DELIMITER ;
```

Execute the stored procedure with ProductID = 680

```
156  CALL GetProductInfo(680);
```

100% 1:156 4 errors found

Result Grid Filter Rows: Search Export:

ProductID	ProductName	ProductDescription
680	HL Road Frame - Black, 58	Replacement mountain wheel for entry-level rider.