Date.....
Page.....

Mission Name: Group Summary Statistics

* CASE Clause:

Syntax:-

```
CASE
WHEN <condition_1> THEN <value_1>
WHEN <condition_2> THEN <value_2>
ELSE <value_3>
END AS <new_column_name>
```

(ELSE ... is optional)

Eg:-

```
SELECT Major, Sample_size,
CASE
WHEN Sample_size < 200 THEN "Small"
WHEN Sample_size >= 200 AND Sample_size < 1000 THEN "Medium"
WHEN Sample_size >= 1000 THEN "LARGE"
END AS "Sample_category"
FROM recent_grads
```

* Group statistics are calculated only for columns in which the value represents categories.

* GROUP BY SQL Statement allows us to do so.

Eg:- SELECT Major_category, SUM(Total) AS Total_graduates
FROM recent_grads GROUP BY Major_category;

* SELECT Major_category, AVG(ShareWomen) AS Average_women
FROM recent_grads GROUP BY Major_category.

* Multiple Summary Statistics By Group:-

Eg:- SELECT Major_category, SUM(Women) AS Total_women,
AVG(ShareWomen) AS Mean_women, SUM(Total)*AVG(Share-
Women) AS Estimate_women FROM recent_grads
GROUP BY Major_category.

* GROUP BY can also be used over multiple columns.

* Aggregate functions cannot be used with GROUP BY.

Eg:-

```
SELECT Major_category, Sample_category,
       AVG(ShareWomen) AS Mean_women,
       SUM(Total) AS Total_graduates
FROM new_grads GROUP BY Major_category,
                        Sample_category;
```

* HAVING should be used with GROUP BY, not WHERE because HAVING also works for virtual column.

→ Eg:-

```
SELECT Major_category, AVG(Low_wage_job)/AVG(Total)
AS Share_low_wage FROM new_grads GROUP BY
Major_category HAVING Share_low_wage > 0.1;
```

* Order of Execution:-

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY
7. LIMIT

* ROUND() Function:-

Syntax:- ROUND(col_name, no. of decimal places)

Eg:- ROUND(ShareWomen, 2)

Eg:-

```
SELECT ROUND(ShareWomen, 4) AS Rounded_women,
       Major_category
FROM new_grads LIMIT 10;
```

Date.....
Page.....

* Nesting Functions:-

```
SELECT Major_category,
       ROUND(AVG(College_job)/AVG(Total), 3) AS
       Share_degree_jobs
FROM new_grads GROUP BY Major_category HAVING
Share_degree_jobs < 0.3
```

* If two columns with integer values are divided, then SQLite will round down and return integer values.

Eg:- 282/2339 will become 0 instead of 0.120.

* To solve the issue we can use CAST() Function.

Syntax:- CAST(col_name AS data_type)

Eg:- CAST(Women AS FLOAT)

Eg:-

```
SELECT Major_category, CAST(SUM(Women) AS FLOAT)/
                       CAST(SUM(Total) AS FLOAT) AS
                       SW
FROM new_grads GROUP BY Major_category
               ORDER BY SW;
```