SQL Session 3

Concepts to be covered:

1) **Aggregation**
2) **GROUP BY**
3) **HAVING Clause**
4) **Conditional Statements**
5) **CASE**

Here is a simplified breakdown of each topic:

1. Aggregation:

Explanation: Aggregation is about combining multiple rows of data to return a single result. **Example:** Using functions like SUM, AVG, MIN, MAX to calculate totals, averages, or extremes from a column of numbers.

2. GROUP BY:

Explanation: GROUP BY divides the rows returned from a query into groups based on a column's values.
Example: If you want to find the total sales for each product category, you'd use GROUP BY on the "category" column.

3. HAVING Clause:

Explanation: HAVING is like WHERE but for groups; it filters grouped data based on conditions.
Example: If you want to see only product categories with total sales greater than 1000, you'd use HAVING.

4. Conditional Statements:

Explanation: These are used to perform different actions based on whether a condition is true or false.
Example: Using IF or CASE statements to make decisions in SQL based on specific conditions.

5. CASE:

Explanation: CASE is a conditional statement used to create different outputs based on different conditions.
Example: Assigning different labels based on sales figures: "High," "Medium," "Low" using CASE.
These topics cover how to aggregate data (like summing or averaging), grouping results,

applying conditions both on individual rows and grouped data, and using conditional statements to manipulate and analyse data within a database.

Datasets used:

- `billing`
- `customer`
- `feedback`
- `service_packages`
- `service_usage`

Example Queries:

Basic:

1.Find the average monthly rate for each service type in service_packages.

SELECT service_type, ROUND(AVG(monthly_rate),2) AS average_monthly_rate FROM service_packages
GROUP BY service_type;

Result Grid | Filter Rows:

service_type	average_monthly_rate
Broadband	54.83
Mobile	55.75
Streaming	53.72

2. Calculate the total minutes used by all customers for mobile services.

SELECT SUM(minutes_used) AS total_mobile_minutes
FROM service_usage
WHERE service_type = 'mobile';

Result Grid | Filter Rows:

total_mobile_minutes
2509255

3. List the total number of feedback entries for each rating

level. *SELECT rating, COUNT(*) AS feedback_count*

FROM feedback

GROUP BY rating;

Result Grid | Filter Rows:

rating	feedback_count
4	194
1	211
2	167
	35
3	203
5	190

Intermediate:

1. Group feedback by service impacted and rating to count the number of feedback entries.

UPDATE feedback SET rating = NULL WHERE rating = '';
UPDATE feedback SET service_impacted = NULL WHERE service_impacted = '';

SELECT service_impacted, rating, COUNT(*) AS feedback_count FROM feedback
GROUP BY service_impacted, rating
ORDER BY service_impacted, rating;

Result Grid | Filter Rows:

service_impacted	rating	feedback_count
		1
	1	6
	2	6
	3	8
	4	3
	5	7
Broadband		9
Broadband	1	71
Broadband	2	51
Broadband	3	64
Broadband	4	68
Broadband	5	53
Mobile		13

2. Calculate the total data and minutes used per customer, per service type.

SELECT customer_id, service_type, SUM(data_used) AS total_data_used, SUM(minutes_used) AS total_minutes_used
FROM service_usage

GROUP BY customer_id, service_type;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	service_type	total_data_used	total_minutes_used
178	Broadband.Streaming	245.03	2097
594	Broadband.Streaming	140.34	6506
61	Mobile	703.56	1034
670	Broadband.Streaming	105.89	674
632	Mobile	219.73	10032
100	Broadband.Streaming	1080.9099999999999	11601
706	Broadband.Streaming	1760.01	16925
466	Broadband.Streaming	939.48	9317
853	Broadband.Streaming	767.05	4831
217	Mobile	45.23	2350
543	Broadband.Streaming	187.78	9828
374	Broadband.Streaming	354.97	7687
562	Broadband.Streaming	937.52	7871

3. Determine which customers have provided feedback on more than one type of service, but have a total rating less than 10.

SELECT customer_id
FROM feedback
GROUP BY customer_id
HAVING COUNT(DISTINCT service_impacted) > 1 AND SUM(rating) <

Result Grid | Filter R

customer_id
1
7
8
13
22
27
32
34
45
52
56
75
77

10;

Hard:

1. Classify each customer as 'High Value' if they have a total amount due greater than

$500, or ‘Standard Value’ if not.

```
SELECT customer_id,
 IF(SUM(amount_due) > 500, 'High Value', 'Standard Value') AS value_category
FROM billing
GROUP BY customer_id;
```

Result Grid | Filter Rows:

customer_id	value_category
100	High Value
101	High Value
102	High Value
103	High Value
104	High Value
105	High Value
106	High Value
107	High Value
108	High Value
109	High Value
110	High Value
111	High Value
112	High Value

2. Update the discounts_applied field in billing to 10% of amount_due for bills with a payment_date past the due_date, otherwise set it to 5%.

```
Set sql_safe_updates=0;

UPDATE billing
SET discounts_applied =
 CASE
 WHEN payment_date > due_date THEN amount_due * 0.1
 ELSE amount_due * 0.05
 END;

SELECT *
FROM billing;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
1	100	8262.08	5/4/2023	12/22/2023	23-Nov	413.10400000000004	955.9
2	101	1694.71	9/17/2023	8/17/2023	23-Jun	84.7355	487.07
3	102	9367.53	11/4/2023	9/25/2023	23-Jan	936.7530000000002	177.2
4	103	1101.08	10/17/2023	11/1/2023	23-Jun	110.108	30.09
5	104	1041.16	10/2/2023	3/13/2023	23-Jan	104.11600000000001	486.27
6	105	6753.59	6/21/2023	1/13/2023	23-Apr	337.6795	846.89
7	106	2737.5	6/26/2023		23-Jun	136.875	605.7
8	107	5566.26	1/28/2023	2/21/2023	23-Jun	556.6260000000001	70.45
9	108	3644.35	6/5/2023	11/1/2023	23-Dec	182.2175	682.41
10	109	9717.41	9/4/2023		23-Feb	485.8705	843.36

3. In billing, create a flag for each bill that is ‘Late’ if the payment_date is after the

due_date, ‘On-Time’ if it’s the same, and ‘Early’ if before.

```
SELECT bill_id,
 CASE
 WHEN payment_date > due_date THEN 'Late'
 WHEN payment_date = due_date THEN 'On-Time'
 WHEN payment_date < due_date THEN 'Early'
 END AS payment_status
FROM billing;
```

Result Grid | Filter Rows:

bill_id	payment_status
1	Early
2	Early
3	Late
4	Late
5	Late
6	Early
7	Early
8	Late
9	Early
10	Early

These queries progress from basic aggregation and grouping to more complex scenarios involving conditional statements, CASE expressions, and data updates based on specific conditions.

Here are some websites that offer tutorials and resources to learn SQL concepts, including aggregation, conditional statements, and more:

https://sqlzoo.net/wiki/SQL_Tutorial

https://www.tutorialspoint.com/sql/index.htm

https://sqlbolt.com/

https://www.w3schools.com/sql/

Happy Learning!