**SQL class 3:**   
  
While loading the data set, make sure to set the date columns as datetime and remove time from the format option.   
 **Topics covered in today’s class**

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

**Aggregating Data**

**Find the average monthly rate for each service type in service\_packages.**

SELECT service\_type, AVG(monthly\_rate) AS average\_monthly\_rate

FROM service\_packages

GROUP BY service\_type;

**Introduce the ROUND function here to make result set more neat**

SELECT service\_type, ROUND(AVG(monthly\_rate),2) AS average\_monthly\_rate

FROM service\_packages

GROUP BY service\_type;

**Identify the customer who has used the most data in a single service\_usage record. ( covers ORDER BY and LIMIT that we did in last class)**

SELECT customer\_id, MAX(data\_used) AS max\_data\_used

FROM service\_usage

GROUP BY customer\_id

ORDER BY max\_data\_used DESC

LIMIT 1;

**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';

**List the total number of feedback entries for each rating level.**

SELECT rating, COUNT(\*) AS feedback\_count

FROM feedback

GROUP BY rating;

**Show the total amount due by each customer, but only for those who have a total amount greater than $100.( introducing HAVING clause here) Using billing table here**

SELECT customer\_id, SUM(amount\_due) AS total\_amount\_due

FROM billing

GROUP BY customer\_id

HAVING total\_amount\_due > 100;

**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) < 10;

**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;

**Conditional Expressions and CASE Statements**

**Categorize customers based on their subscription date: ‘New’ for those subscribed after 2023-01-01, ‘Old’ for all others.**

SELECT customer\_id,

CASE

WHEN subscription\_date > '2023-01-01' THEN 'New'

ELSE 'Old'

END AS customer\_type

FROM customer;

**In service\_usage, label data usage as ‘High’ if above the average usage, ‘Low’ if below.**

SELECT usage\_id, data\_used,

CASE

WHEN data\_used > (SELECT AVG(data\_used) FROM service\_usage) THEN 'High'

ELSE 'Low'

END AS usage\_level

FROM service\_usage;

**For each feedback given, categorise the service\_impacted into ‘Digital’ for ‘streaming’ or ‘broadband’ and ‘Voice’ for ‘mobile’.**

SELECT feedback\_id, service\_impacted,

CASE service\_impacted

WHEN 'streaming' THEN 'Digital'

WHEN 'broadband' THEN 'Digital'

WHEN 'mobile' THEN 'Voice'

END AS service\_category

FROM feedback;

**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%.**

UPDATE billing

SET discounts\_applied =

CASE

WHEN payment\_date > due\_date THEN amount\_due \* 0.1

ELSE amount\_due \* 0.05

END;

**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;

**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;

**EXTRA EXERCISES:**

Data Cleaning:

Identify customers with a phone number that does not start with "555".

SELECT \* FROM customer WHERE phone\_number NOT LIKE '555%';

Replace any occurrences of "Road" in the address field with "Rd.".

UPDATE customer SET address = REPLACE(address, 'Road', 'Rd.');

Convert the billing cycle in the `billing` table to uppercase.

UPDATE billing SET billing\_cycle = UPPER(billing\_cycle);

Identify any records in the `billing` table where the `discounts\_applied` is negative.

SELECT \* FROM billing WHERE discounts\_applied < 0;

Remove any leading or trailing whitespaces from the `name` field in the `customer` table.

UPDATE customer SET first\_name = TRIM(first\_name);

Data Transformation:

Add a month to all the `subscription\_date`s in the `customer` table.

UPDATE customer SET subscription\_date = DATE\_ADD(subscription\_date, INTERVAL 1 MONTH);

Extract the year from the `subscription\_date` for all customer and display their `name` alongside.

SELECT name, YEAR(subscription\_date) AS 'Subscription Year' FROM customer;

Concatenate the `name` and `email` fields, separated by a colon, and display the result for all customer.

SELECT CONCAT(first\_name, ': ', email) AS 'Name and Email' FROM customer;