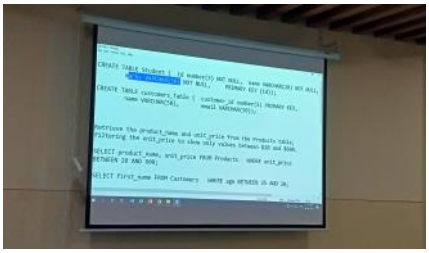
Lab - 04

Friday, September 6, 2024 11:21 AM

```
CREATE TABLE student ( id number(3) NOT NULL, name VARCHAR(10) NOT NULL ,marks VARCHAR(5), PRIMARY KEY (id));
CREATE TABLE customers_Table ( customer_id number(3) PRIMARY KEY,name VARCHAR(10), email VARCHAR(10));

VARCHAR(50));
Retrieve the product_name and unit_price from the Products table,
filtering the unit_price to show only values between $20 and $600.
SELECT product_name, unit_price FROM Products WHERE unit_price
BETWEEN 20 AND 600;
SELECT first name FROM Customers WHERE age BETWEEN 25 AND 30;
```



It looks like you have a mix of SQL queries with some minor issues. Here's a corrected version for each section:

1. Creating the student table:

sql
Copy code
```
CREATE TABLE student (
   id NUMBER(3) NOT NULL,
   name VARCHAR(10) NOT NULL,
   marks VARCHAR(5),
   PRIMARY KEY (id)
);
```

2. Creating the customers_Table table:

sql
Copy code
```
CREATE TABLE customers_Table (
   customer_id NUMBER(3) PRIMARY KEY,
   name VARCHAR(10),
   email VARCHAR(50) -- You had a misplaced `VARCHAR(50)`; corrected here.
);
```

3. Retrieving the product name and unit price from the Products table with filtering:

sql
Copy code
```
SELECT product_name, unit_price FROM Products WHERE unit_price BETWEEN 20 AND 600;
```

4. Selecting first_name from Customers table based on age filtering:

You missed the underscore in first_name, and the column age should also be part of your Customers table structure.

sql
Copy code
```
SELECT first_name
FROM Customers
WHERE age BETWEEN 25 AND 30;
```

These should now work correctly assuming your table structures and column names are as expected.

1. Creating a Products table:

You can create a Products table as follows:

sql
Copy code
```
CREATE TABLE Products (product_id NUMBER(5) PRIMARY KEY,product_name VARCHAR(15) NOT NULL,unit_price NUMBER(4));
```

2. Inserting values into the Products table:

1. Inserting individual values into the Products table:

sql
Copy code
```
INSERT INTO Products (product_id, product_name, unit_price) VALUES (1001, 'Laptop', 450.50);
```

sql
Copy code
```
INSERT INTO Products (product_id, product_name, unit_price) VALUES (1002, 'Smartphone', 300.75);
```

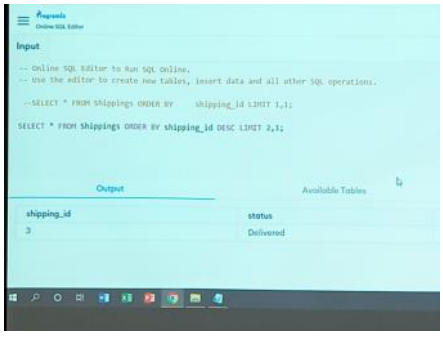
sql
Copy code
```
INSERT INTO Products (product_id, product_name, unit_price) VALUES (1003, 'Tablet', 150.25);
```

sql
Copy code
```
INSERT INTO Products (product_id, product_name, unit_price) VALUES (1004, 'Headphones', 50.00);
```

sql
Copy code
```
INSERT INTO Products (product_id, product_name, unit_price) VALUES (1005, 'Monitor', 200.00);
```

3. Query to retrieve product_name and unit_price where unit_price is between $20 and $600:

sql
Copy code
```
SELECT product_name, unit_price
FROM Products
WHERE unit_price BETWEEN 20 AND 600;
```



Second , third min and max value find krna ::::::

Step 1: Creating a Table

sql
Copy code
```
CREATE TABLE Shippings (shipping_id NUMBER PRIMARY KEY, shipping_date DATE,shipping_status VARCHAR2(20));
```

Step 2: Inserting Data

sql
Copy code
```
INSERT INTO Shippings (shipping_id, shipping_date, shipping_status)VALUES (1, TO_DATE('2024 -09-01', 'YYYY-MM-DD'), 'Shipped');
INSERT INTO Shippings (shipping_id, shipping_date, shipping_status)VALUES (2, TO_DATE('2024 -09-02', 'YYYY-MM-DD'), 'Pending');
INSERT INTO Shippings (shipping_id, shipping_date, shipping_status)VALUES (3, TO_DATE('2024 -09-03', 'YYYY-MM-DD'), 'Delivered');
```

Step 3: Select with ORDER BY and LIMIT

To get the first row based on shipping_id:

sql
Copy code
```
SELECT *FROM Shippings ORDER BY shipping_id OFFSET 1 ROWS FETCH NEXT 1 ROWS ONLY;
```
To get the second last row based on shipping_id in descending order:

sql
Copy code
```
SELECT *
FROM Shippings
ORDER BY shipping_id DESC
OFFSET 1 ROWS FETCH NEXT 1 ROWS ONLY;
```

Step 1: Fetching Rows Using ROWNUM

To get the second row (offset 1) in ascending order of shipping_id:

sql
Copy code
```
SELECT *FROM (SELECT shipping_id, shipping_date,shipping_status
FROM Shippings ORDER BY shipping_id) WHERE ROWNUM = 2;
```

Step 2: Fetching the Second-Last Row in Descending Order

To get the second-last row in descending order of shipping_id:

sql
Copy code
```
SELECT *
FROM (
   SELECT shipping_id, shipping_date, shipping_status
   FROM Shippings
   ORDER BY shipping_id DESC
)
WHERE ROWNUM = 2;
```

For Oracle, you cannot use LIMIT directly. To achieve the same functionality, you can use the ROWNUM or ROW_NUMBER() functions, or the OFFSET and FETCH clauses if you're on Oracle 12c or later.
Here's the correct approach based on your request:

1. Fetching the 2nd row (equivalent to LIMIT 1,1):

Since Oracle doesn't directly support LIMIT, we can use ROWNUM or ROW_NUMBER() for such queries.
For the second row (like LIMIT 1,1 in MySQL):

sql
Copy code
```
SELECT * FROM ( SELECT shipping_id, shipping_date, shipping_status, ROW_NUMBER() OVER (ORDER BY shipping_id) AS rn  FROM Shippings) WHERE rn = 2;
```

2. Fetching the 2nd last row in descending order (like LIMIT 2,1):

For the second row when ordered in descending order by shipping_id, we again use ROW_NUMBER():

sql
Copy code
```
SELECT *
FROM (
   SELECT shipping_id, shipping_date, shipping_status, ROW_NUMBER() OVER (ORDER BY shipping_id DESC) AS rn
   FROM Shippings
)
WHERE rn = 2;
```