

Question 1

1.1) The error appears because the table that is referenced is from a schema that was not created by me but is from another schema. To correct the error, the query should reference the table by the schema name.

1.2) SELECT Last_Name, Year_Admission

From studentlist

WHERE Last_Name LIKE '%y' AND Year_Admission >= 2020;

SQL Worksheet

```
1 SELECT Last_Name, Year_Admission
2 From studentlist
3 WHERE Last_Name LIKE '%y' AND Year_Admission >= 2020;
```

1.3 SELECT DISTINCT Last_Name FROM studentlist;

SQL Worksheet

```
1 SELECT DISTINCT Last_Name FROM studentlist;
```

1.4 SELECT distinct First_Name, Last_Name FROM `studentlist`;

SQL Worksheet

```
1 SELECT distinct First_Name, Last_Name FROM `studentlist`;
```

1.5 SELECT * FROM studentlist

WHERE Last_Name IN ('Evelyn', 'Samson', 'Paul');

SQL Worksheet

```
1 SELECT * FROM studentlist
2 WHERE Last_Name IN ('Evelyn', 'Samson', 'Paul');
```

Question 2

2.1) INSERT INTO studentlist (Last_Name)

VALUES ('Johnson');

SQL Worksheet

```
1 INSERT INTO studentlist (Last_Name)
2 VALUES ('Johnson');
```

2.2) UPDATE stafflist

SET Salary = '+2000'

WHERE CompanyID = 02361;

SQL Worksheet

```
1 UPDATE stafflist
2 SET Salary = '+2000'
3 WHERE CompanyID = 02361;
```

2.3) You should enclose an alias in double quotes for when there are special characters, spaces, reserved keyword and for punctuation.

Question 3

3.1) A join is a feature of SQL that allows you to view the columns from two or more tables in a single query. Separate tables are used in databases to hold data, with each table housing a particular kind of data. You can create a query that pulls data from multiple sources using a JOIN function. A join in SQL involves a few things:

- Two tables
- A JOIN keywords
- Conditions that specify how the two tables are joined

JOIN Syntax:

SELECT columns

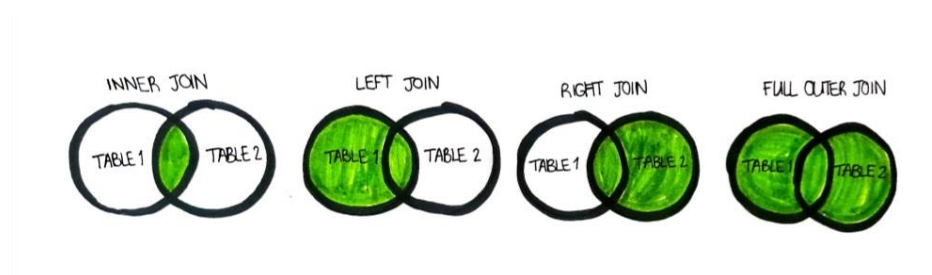
FROM table1

JOIN table2 ON criteria;

3.2) There are 4 types of JOIN's in SQL. These return records from tables including:

- Inner JOIN: Includes matching values in both tables.
- LEFT (OUTER) JOIN: Includes all records from left table, and the matching records from the right table
- RIGHT (OUTER) JOIN: Includes all records from right table, and the matching records from the left table
- FULL (OUTER) JOIN: Returns all the records from the match left and right table.

Diagrams to give an explanation:

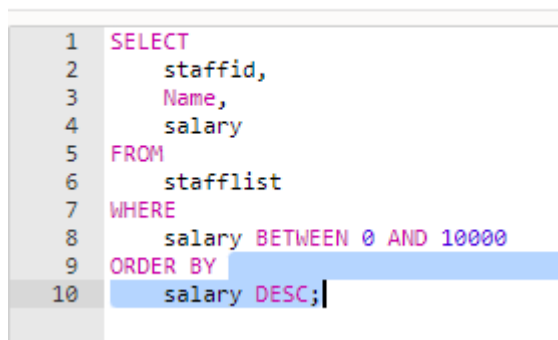


Question 4

4.1) Low Wage Query:

```
SELECT
    staffid,
    Name,
    salary
FROM
    stafflist
WHERE
    salary BETWEEN 0 AND 10000
ORDER BY
    salary DESC;
```

SQL Worksheet

A screenshot of an SQL Worksheet interface. It features a list of line numbers from 1 to 10 on the left side. The corresponding SQL query is entered on the right. The query is: SELECT staffid, Name, salary FROM stafflist WHERE salary BETWEEN 0 AND 10000 ORDER BY salary DESC;. The text is color-coded: keywords like SELECT, FROM, WHERE, ORDER BY, and DESC are in pink, while identifiers like staffid, Name, salary, stafflist, and values like 0, 10000 are in black. The line numbers 1 through 10 are in a light grey box on the left.

```
1 SELECT
2     staffid,
3     Name,
4     salary
5 FROM
6     stafflist
7 WHERE
8     salary BETWEEN 0 AND 10000
9 ORDER BY
10    salary DESC;
```

Save table as Low_Wage_Table

Meduim wage Query:

```
SELECT
    staffid,
    Name,
    salary
FROM
    stafflist
WHERE
    salary BETWEEN 10000 AND 20000
ORDER BY
    salary DESC;
```

SQL Worksheet

```
1 SELECT
2     staffid,
3     Name,
4     salary
5 FROM
6     stafflist
7 WHERE
8     salary BETWEEN 10000 AND 20000
9 ORDER BY
10    salary DESC;
```

Save Table as Medium_Wage

High Wage Query:

```
SELECT
    staffid,
    Name,
    salary
FROM
    stafflist
WHERE
    salary >= 20000
ORDER BY
    salary DESC;
```

SQL Worksheet

```
1 SELECT
2     staffid,
3     Name,
4     salary
5 FROM
6     stafflist
7 WHERE
8     salary >= 20000
9 ORDER BY
10    salary DESC;
```

Save Table as High_Wage

Query to combine tables SELECT Name, salary ,

CASE WHEN salary < 10000 THEN 'Low Wage' WHEN salary >= 10000 AND salary < 20000 THEN 'Medium Wage' ELSE 'High Wage' END AS salary_category FROM stafflist;

SQL Worksheet

```
1 SELECT Name, salary ,
2 CASE WHEN salary < 10000 THEN 'Low Wage' WHEN salary >= 10000 AND salary < 20000 THEN 'Medium Wage' ELSE 'High Wage' END AS salary_category FROM stafflist;
3
```

4.2) SELECT staffid, Name, Wage, Address

FROM stafflist

WHERE UPPER(Name) = 'JOHNSON' OR LOWER(Name) = 'johnson';

SQL Worksheet

```
1 SELECT staffid, Name, Wage, Address
2 FROM stafflist
3 WHERE UPPER(Name) = 'JOHNSON' OR LOWER(Name) = 'johnson';
```

Question 5

5.1) SELECT Emp_Name,EMP_ID, JOB_ID,Tasks_Completed, Year, COUNT(*) AS NoOfEmployeesLeft
SUM (EMP_ID)

FROM Employee

Group BY Year, JOBID

ORDER BY Year DESC, NoOfEmployeesLeft DESC;

SQL Worksheet

```
1 SELECT Emp_Name,EMP_ID, JOB_ID,Tasks_Completed, Year, COUNT(*) AS NoOfEmployeesLeft
2 SUM (EMP_ID)
3 FROM Employee
4 Group BY Year, JOBID
5 ORDER BY Year DESC, NoOfEmployeesLeft DESC;
```

5.2) Yes you can count the records. By dividing the records in their own separate category each record can be counted. SQL will help organize the data by running the query to divide employee records by year of employment, then by job, and by salary. Each category can be counted with the query.

SQL Worksheet

```
1 SELECT Year, Job, Salary
2 FROM Employee
3
```

5.3) To run a script on SQL*PLUS you can copy and paste or enter the file's path and run it in SQL*Plus. Type the following code:

SQL > @{File}

For example if the file is already in the directory:

```
SQL > @thescript.sql
```

If the file is not in the directory:

```
SQL > @{path}{File}
```

```
SQL > @/oracle/myscripts/thescript.sql
```