All Possible generic SQL statements

1. To view an entire table

SELECT \*

FROM <Table\_name>;

1. To view select rows from a table

SELECT \*

FROM <Table\_name>

WHERE <Conditions>;

1. To Insert new entries into a table

INSERT INTO <Table\_name(attr1, attr2, ...)>

VALUES <Values of each attribute>;

1. To update existing entry from a table

UPDATE <Table\_name>

SET <attribute = value>

WHERE <Conditions>;

1. To delete existing entry from a table

DELETE FROM <Table\_name>

WHERE <Conditions>;

All the SQL Statements we have will follow one of the 5 forms described above. We need to replace the <...> with real values case by case as follows:

* For number 1, to view an entire table, to view specific attributes from the table, we can change \* with select attributes that we wanted and <Table\_name> will be replaced with the actual table name.
* For number 2, to view select rows from the table, similar to number 1, we can replace \* with select attributes we wanted and <Table\_name> will be replaced with an actual table name. Moreover, <Conditions> will be replaced with 1 or more conditions to filter data. For instance if we want only Employees of age 30 and above, we replace <Conditions> with age >= 30 where age is the attribute name.
* For number 3, to insert new entries into a table, we replace <Table\_name(attr1, attr2 ,…)> with an actual table name followed by the attributes of the relation inside the brackets and <Values of each attribute> with be replaced with actual attribute values (separated by commas) with respect to the attribute names.
* For number 4, to update an existing entry from a table, we replace <Table\_name> with the table name, <attribute = value> with an attribute name and its value, we may repeat this statement (separated by commas) and <Conditions> will be replaced with 1 or more conditions to filter which data entries we will be updating.
* For number 5, To delete existing entry from a table, we replace <Table\_name> with the table name and <Conditions> will be replaced with 1 or more conditions to filter which data entries will be deleted.

Below are the list of <Table\_name> and its attributes:

* Employee (Emp\_id, Fname, Lname, SEmp\_id)
* Supervisor (SEmp\_id)
* BackEmployee (BEmp\_id)
* FrontEmployee (FEmp\_id)
* Family (Cli\_id)
* Child (Cli\_id, Name, Cals\_needed, Age, Gender)
* Adult (Cli\_id, Name, Cals\_needed, Age, Gender)
* Order (Order\_no, Picked\_up, BEmp\_id)
* F\_Order (FOrder\_no)
* C\_Order (COrder\_no)
* Food\_Inventory (Name, Qty)
* Food (Name, Id, Type, Desc, Calories, FOrder\_no)
* Replenishes\_F (SEmp\_id, Name)
* F\_Supplies (FOrder\_no, Name)
* Clothing\_Inventory (Type, Size, Gender, Qty)
* Clothes (Type, Size, Gender, Id, Condition, Brand, COrder\_no)
* Replenishes\_C (SEmp\_id, Type, Size, Gender)
* C\_Supplies (COrder\_no, Type, Size, Gender)
* Orders (FEmp\_id, Order\_no, Cli\_id, Date, Time)

Below are some of the example queries, we will not be covering for all the tables since some will be similar to others to avoid repititions

For Employee tables,

1. View Employee table:

SELECT \*

FROM Employee;

1. View certain employee:

SELECT \*

FROM Employee

WHERE Emp\_id = 3;

1. Insert New Employee:

INSERT INTO Employee (Emp\_id, Fname, Lname, SEmp\_id)

VALUES 3,”Steven”, “Susanto”, 1;

1. Update an employee:

UPDATE Employee

SET Lname = “Smith”

WHERE Emp\_id = 3;

1. Delete existing employee from Employee Table:

DELETE FROM Employee

WHERE Emp\_id = 3;

For Child and Adult tables, here the Adult and Child are interchangable

1. View Adult table:

SELECT \*

FROM Adult;

1. View certain Adult:

SELECT \*

FROM Adult

WHERE Cli\_id = 1 AND Name = “John Doe”;

1. Insert New Adult:

INSERT INTO Adult (Cli\_id, Name, Cals\_needed, Age, Gender)

VALUES 1,”John Doe”, 2500, 25, “Male”;

1. Update an Adult entry:

UPDATE Adult

SET Cals\_needed = 3000

WHERE Cli\_id = 1 AND Name = “John Doe”;

1. Delete existing adult from Adult Table:

DELETE FROM Adult

WHERE Cli\_id = 1 AND Name = “John Doe”;

For Order table,

1. View Order table:

SELECT \*

FROM Order;

1. View Orders made by a specific back employee:

SELECT \*

FROM Order

WHERE BEmp\_id = 10;

1. Insert New Order:

INSERT INTO Order (Order\_no, Picked\_up, BEmp\_id)

VALUES 112, ”F”, 10;

1. Update an Order entry:

UPDATE Order

SET Picked\_up = “T”

WHERE Order\_no = 112;

1. Delete existing order from Order Table:

DELETE FROM Order

WHERE Order\_no = 112;

For Food\_inventory table,

1. View Food\_inventory table:

SELECT \*

FROM Food\_inventory;

1. View specific items from Food\_inventory:

SELECT \*

FROM Food\_inventory

WHERE Name = “Potato”;

1. Insert New item into Food\_inventoryr:

INSERT INTO Food\_inventory (Name,Qty)

VALUES “Potato”, 100;

1. Update an Food\_inventory entry:

UPDATE Food\_inventory

SET Qty = 30

WHERE Name = “Potato”;

1. Deleting does not make sense here, for instance when stock runs out if we delete, when it is restocked, the Employee would need to make a new table that previously existed.

For Orders table

1. View Orders table:

SELECT \*

FROM Orders;

1. View Orders made by a specific back employee:

SELECT \*

FROM Orders

WHERE BEmp\_id = 10;

1. Insert New Orders:

INSERT INTO Orders (FEmp\_id, Order\_no, Cli\_id, Date, Time)

VALUES 112, ”F”, 10;

1. Update an Orders entry:

UPDATE Orders

SET Picked\_up = “T”

WHERE Order\_no = 112;

1. Delete existing order from Orders Table:

DELETE FROM Order

WHERE Order\_no = 112;