Query single table

Query data in columns c1, c2 from a table
```
SELECT c1, c2 FROM t;
```

Query all rows and columns from a table
```
SELECT * FROM t;
```

Query data and filter rows with a condition
```
SELECT c1, c2 FROM t
WHERE condition;
```

Query distinct rows from a table
```
SELECT DISTINCT c1 FROM t
WHERE condition;
```

Sort the result set in ascending or descending order
```
SELECT c1, c2 FROM t
ORDER BY c1 [ASC | DESC];
```

Skip offset of rows and return the next n rows
```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```

Group rows using an aggregate function
```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```

Filter groups using HAVING clause
```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```

SQL Aggegate functions

- **AVG** returns the average of a list
- **COUNT** returns the number of elements of a list
- **SUM** returns the total of a list
- **MAX** returns the maximum value in a list
- **MIN** returns the minimum value in a list

Joins

Inner join t1 and t2
```
SELECT c1, c2
FROM t1
JOIN t2 ON condition;
```

Left join t1 and t2
```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```

Right join t1 and t2
```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```

Perform full outer join
```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```

Perform a cross join / cartesian product
```
SELECT c1, c2
FROM t1, t2;
```

Join t1 to itself using JOIN clause
```
SELECT c1, c2
FROM t1 A
JOIN t1 B ON condition;
```

SQL Operators

Combine rows from two queries
```
SELECT c1, c2 FROM t1
UNION
SELECT c1, c2 FROM t2;
```

Return the intersection of two queries
```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```

Subtract a result set from another result set
```
SELECT c1, c2 FROM t1
EXCEPT
SELECT c1, c2 FROM t2;
```

Query rows using pattern matching
```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```

Query rows in a list
```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```

Query rows between two values
```
SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND hi;
```

Check if value in a table is NULL or not
```
SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
```

Managing Tables

Create a new table with three columns
```
CREATE TABLE t (
id SERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
price NUMERIC(10,2) DEFAULT 0
);
```

Delete the table from the database
```
DROP TABLE t;
```

Add a new column c to the table
```
ALTER TABLE t ADD column c type;
```

Drop column c from the table
```
ALTER TABLE t DROP COLUMN c;
```

Remove all data in a table
```
TRUNCATE TABLE t;
```

Using SQL constraints

Set c1 and c2 as a primary key
```
CREATE TABLE t(
c1 INT, c2 INT, c3 VARCHAR,
PRIMARY KEY (c1,c2)
);
```

Set c2 column as a foreign key
```
CREATE TABLE t1(
c1 SERIAL PRIMARY KEY,
c2 INT,
FOREIGN KEY (c2) REFERENCES t2(c2)
);
```

Make the values in c1 and c2 unique
```
CREATE TABLE t(
c1 INT, c1 INT,
UNIQUE(c2,c3)
);
```

Ensure c1 > 0 and values in c1 >= c2
```
CREATE TABLE t(
c1 INT, c2 INT,
CHECK(c1> 0 AND c1 >= c2)
);
```

Set values in c2 column not NULL
```
CREATE TABLE t(
c1 SERIAL PRIMARY KEY,
c2 VARCHAR NOT NULL
);
```

Stored Procedures

Basic template for stored procedure
```
CREATE OR REPLACE FUNCTION
  func_name(parameters) RETURNS
  data_type AS $$
DECLARE
  variable_declarations
BEGIN
  function_body
END; $$ LANGUAGE plpgsql;
```

Assignment uses :=
```
x := y + z
```

Fetch single value from sql query
```
SELECT single_value INTO var FROM
  table WHERE id = 17;
```

Trigger

Create or modify a trigger
```
CREATE OR REPLACE TRIGGER
  trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE FUNCTION stored_procedure;
```

WHEN:
BEFORE – invoke before the event occurs
AFTER – invoke after the event occurs
EVENT:
INSERT – invoke for INSERT
UPDATE – invoke for UPDATE
DELETE – invoke for DELETE
TRIGGER_TYPE:
FOR EACH ROW
FOR EACH STATEMENT

Create a trigger invoked before a new row is inserted into the person table
```
CREATE TRIGGER
  before_insert_person
BEFORE INSERT
ON person FOR EACH ROW
EXECUTE FUNCTION stored_procedure;
```

Delete a specific trigger
```
DROP TRIGGER trigger_name;
```