CSCE 156 – SQL Supplemental Example Sheet

Query Type	Syntax Example	Notes
Simple SELECT query	`SELECT * FROM Albums;`	Returns all columns of all records in the Albums table.
Simple SELECT query with column aliases	`SELECT AlbumTitle AS Title, AlbumId AS Id FROM Albums;`	Selects only the two specified columns, aliased (renamed) as shown from all records in the albums database
Simple SELECT query with a WHERE clause	`SELECT AlbumTitle FROM Albums WHERE AlbumYear > 2000;`	Selects the title and album id of all albums created after the year 2000
Boolean operators	`SELECT AlbumTitle FROM Albums WHERE AlbumYear > 2000 AND AlbumNumber = 1;`	Selects the title of all albums which where a bands first release after the year 2000
LIKE operator and %	`SELECT AlbumTitle FROM Albums WHERE AlbumTitle LIKE '%fire%'`	Select all album titles which contain the word "fire"
Like operator and _	`SELECT BandName FROM Bands WHERE BandName LIKE '_2'`	Select all band names which have exactly 2 letters where the second letter is "2"
Simple SELECT query with ORDER BY	`SELECT AlbumTitle, AlbumYear FROM Albums ORDER BY AlbumTitle;`	Selects the title and creation year of all albums alphabetically ordered by title
COUNT function	`SELECT COUNT(*) FROM Albums`	Returns the number of albums in the database
COUNT function with WHERE clause	`SELECT COUNT(*) FROM Albums  WHERE AlbumYear  > 2000;`	Returns the number of albums created after the year 2000
AVG function with WHERE clause	`SELECT AVG(TrackLength) from AlbumSongs WHERE TrackNumber =1`	Returns the average length of the first track of all albums
Simple JOIN	`SELECT Bands.BandName, Albums.AlbumTitle FROM Bands  JOIN Albums ON Bands.BandID = Albums.BandID;`	Selects band names and the bands associated album titles
Simple JOIN with table aliases	`SELECT B.BandName, A.AlbumTitle FROM Bands  AS B JOIN Albums AS A ON B.BandID = A.BandID;`	Selects band names and the bands associated album titles

Simple JOIN with table aliases	`SELECT S.SongTitle, A.TrackNumber, A.TrackLength FROM Songs AS S JOIN AlbumSongs AS A ON S.SongID =A.SongID;`	Selects the title, length and track number of all songs which appear in an album
Simple JOIN with table aliases with WHERE clause	`SELECT S.SongTitle, A.TrackNumber, A.TrackLength FROM Songs AS S JOIN AlbumSongs AS A ON S.SongID =A.SongID WHERE A.TrackLength < 60;`	Selects the title, length and track number of all songs which appear in an album that are shorter than 1 minute
Simple JOIN with table aliases with WHERE and ORDER BY clauses	`SELECT S.SongTitle, A.TrackNumber, A.TrackLength FROM Songs AS S JOIN AlbumSongs AS A ON S.SongID =A.SongID WHERE A.TrackLength < 60 ORDER BY S.SongTitle;`	Selects the title, length and track number of all songs which appear in an album that are shorter than 1 minute sorted alphabetically by title
Multiple JOINs with WHERE clause	`SELECT S.SongTitle, A.TrackNumber, A.TrackLength FROM Songs AS S JOIN AlbumSongs AS A ON S.SongID =A.SongID JOIN Albums as T ON A.AlbumID = T.AlbumID WHERE T.AlbumTitle = "Nevermind";`	Selects the title, length and track number of all songs which appear in the album "Nevermind"
Multiple JOINs with WHERE clause	`SELECT S.SongTitle, Als.TrackNumber, Als.TrackLength FROM Songs AS S JOIN AlbumSongs AS Als ON S.SongID =Als.SongID JOIN Albums as A ON Als.AlbumID = A.Album JOIN Bands as B ON A.BandID =B.BandID WHERE B.BandName = "t.A.T.u.";`	Selects the title, length and track number of all songs by "t.A.T.u"
Left (outer) JOIN	`SELECT * FROM Musician m LEFT JOIN BandMusicians bm ON m.MusicianId = bm.MusicianId LEFT JOIN Bands b on b.BandID = bm.BandID`	Selects all musicians along with the bands they are associated with *including* musicians that are not members of any band
Conceptual SELECT with GROUP BY clause	`SELECT AlbumTitle  FROM Albums GROUP BY AlbumYear;`	Selects the title of one album created in each year in no assured order
Simple aggregate function with GROUP BY clause	`SELECT AlbumYear, COUNT(*) AS NumberOfAlbums FROM Albums GROUP BY AlbumYear;`	Returns a list of years and the corresponding number of albums created in each

Simple aggregate function with JOIN and GROUP BY clauses	`SELECT S.SongTitle, COUNT(*) AS NumberOf Versions FROM Songs AS S JOIN AlbumSongs AS A ON S.SongID =A.SongID GROUP BY S.SongID;`	Returns a list of song titles and the number of versions of each song
Aggregate function conditions	`SELECT A.AlbumTitle, AVG(Als.TrackLength) as AvgTrackLength FROM Albums AS A JOIN AlbumSongs AS Als ON A.AlbumID = Als.AlbumID GROUP BY A.AlbumID HAVING AVG(Als.TrackLength) > 360;`	List the titles of all albums having an average track length longer than 6 minutes
Aggregate function conditions	`SELECT A.AlbumTitle, COUNT(Als.TrackLength) as NumberOfTracks FROM Albums AS A JOIN AlbumSongs AS Als ON A.AlbumID = Als.AlbumID GROUP BY A.AlbumID HAVING COUNT(*) > 10;`	List the titles of all albums containing more than 10 tracks
GROUP BY multiple columns with aggregate function condition	`SELECT B.BandName, A.AlbumYear, COUNT(*) FROM Bands AS B JOIN Albums AS A ON B.BandID = A.BandID GROUP BY B.BandID, A.AlbumYear HAVING COUNT(*) > 1;`	List all band names which released more than 1 album in a year
SELECT queries in SELECT queries	`SELECT S.SongTitle, A.TrackLength FROM Songs AS S JOIN AlbumSongs AS A ON S.SongID = A.SongID WHERE A.TrackLength = (SELECT MAX(A.TrackLength) FROM AlbumSongs AS A);`	List all song titles with the maximal track length
SELECT queries in SELECT queries with temporary tables	`SELECT B.BandName FROM Bands AS B JOIN BandMusicians AS BM ON B.BandID = BM.BandID GROUP BY B.BandID HAVING COUNT(*) > (SELECT AVG(T.NumberOfMusicians) FROM (SELECT COUNT(*) AS NumberOfMusicians FROM BandMusicians AS BM GROUP BY BM.BandID) AS T);`	List all band names where the band size is larger than average
SELECT queries in SELECT queries with temporary tables	`SELECT A.AlbumTitle, A.AlbumYear, T.BandName FROM Albums AS A JOIN (SELECT B.BandID, B.BandName, A.AlbumYear FROM Bands AS B JOIN Albums AS A ON B.BandID = A.BandID GROUP BY B.BandID, A.AlbumYear HAVING COUNT(*) > 1) AS T ON T.BandID = A.BandID AND T.AlbumYear = A.AlbumYear;`	Lists the title, year, and corresponding band name of albums released in the same year by the same band

Query Type	Syntax Example	Notes
Simple INSERT	`INSERT INTO Songs VALUES (314159265, ‘Canon’);`	Inserts ‘Canon’ associated with the SongID, 314159265 into Songs
Simple INSERT with specified columns	`INSERT INTO Songs (SongTitle) VALUES (‘Passacaglia’);`	Inserts ‘Passacaglia’ associated with the default auto-constructed SongID into Songs
SET variable and LAST_INSERT_ID	`SET @songID = LAST_INSERT_ID();`	Stores the last inserted primary key id into a newly declared variable songID; note: this is *not standard SQL*
Simple INSERT with a variable	`INSERT INTO Albums (AlbumTitle, AlbumYear, BandID) VALUES (‘Classical Music’, 2012, @bandID);`	Insert using an album with the title ‘Classical Music’ the release date 2012 and the defined BandID; note: this is *not standard SQL*
INSERT with a SELECT query	`INSERT INTO AlbumSongs (SongID, AlbumID) SELECT S.SongID, A.AlbumID  FROM Songs AS S JOIN Albums AS A ON S.SongTitle = 'Passacaglia' AND A.AlbumTitle = 'Classical Music';`	Selects the correct AlbumID and SongID to connect in the AlbumSongs table

Query Type	Syntax Example	Notes
Simple UPDATE	`UPDATE Albums SET AlbumYear = 2011 WHERE AlbumTitle = 'Classical Music';`	Updates the release date of the album titled ‘Classical Music’ to 2011
Simple UPDATE	`UPDATE Albums SET AlbumYear = AlbumYear – 1;`	Reduced the release dates of *all* albums by 1 year

Query Type	Syntax Example	Notes
Simple DELETE	`DELETE FROM Songs WHERE SongID = 314159265;`	Inserts ‘Canon’ associated with the SongID, 314159265 into Songs
Simple DELETE	`DELETE FROM Albums WHERE Albums.AlbumTitle = ‘Classical Music’ AND Albums.AlbumYear = 2012;`	Deletes all albums titled ‘Classical Music’ released in 2012
Simple DELETE of everything in a table (careful!)	`DELETE FROM Songs;`	Deletes all records in the Songs table